blender art MAGAZINE

CHARACTER MODELING !!

blender learning made easy

- Character Modeling at Plumiferos !!
- Modeling a Butterfly
- Rapid Prototyping in Blender
- Blender2Pov
- Character Design 2d sketch to 3d
- ResPower Super/Farm
- MakeHuman

EDITOR/DESIGNER
Gaurav Nawani *gaurav@blenderart.org*

MANAGING EDITOR
Sandra Gilbert *sandra@blenderart.org*

WEBSITE
Nam Pham *nam@blenderart.org*

PROOFER
Kenron Dillon

WRITERS
Christian Guckelsberger
Manuel Perez
Claudio Andaur
José Mauricio Rodas R.
Claas Eicke Kuhnen
Rogério Perdiz
Manuel Bastioni
Alessandro Proglio
Antonio Di Cecca Giovanni
Lanza Martin
Ed Cicka

Sandra Gilbert
Managing Editor

Character modeling has got to be one of my favorite things to do in Blender. There is nothing like starting with a blank screen and ending up with living, moving characters. Now granted, not all my characters turn out as good as I would like, but that just keeps me trying. And there are so many things to try.

One of the best things about character modeling is that a character can be anything. The most common ones are animals, aliens, robots and humanoids, but those kinds of characters are but a small selection of what is possible. With a little work and imagination, you can turn any everyday object into a character. As soon as it starts moving and showing personality, it becomes a character. Television commercials are full of such amusing characters.

As much fun as character modeling can be, it can also be frustrating at times. Achieving a good model that poses well and conveys a sense of life and personality can be an elusive goal. Lighting and texturing often play a big role in helping bring a character to life, as well as a good understanding of how a character should move and the timing involved for believable movement.

In this issue, we will be looking at a couple of characters for you to model and play with, as well as a very informative article on how to take your model from the computer screen to a real-world physical 3D statue! (Wouldn't I just love to have one of those machines at my house. J) Additionally, Malefico Andauer gives us a behind-the-scenes look at some of the decisions made for the Plumiferos Characters.
So sit back, grab a cup of coffee and read all about it ■

- *sandra@blenderart.org*

- Izzy

Character Modeling & Design: *A short run-down of things to consider before you start.*

Just where and how do you start designing a character? Of course, there is always the "just jump right in and do it" method. Which sometimes works wonderfully, but all too often will cause future problems depending on what you had planned to do with your character. Before you open Blender, it is always a good idea to sit down and think about your character. Here is a list of questions you might want to consider before you actually start modeling:

• What style of character are you going for? Realistic, Semi-realistic, Toon.
• What kind of character? Fantasy, Mechanical, Animal, Humanoid, etc...
• Appearance: How many arms, legs, eyes, ears, etc... does it have any at all?
• How will your character interact with it's environment? Does it have/need clothes?
• How does it move?
• How do you plan on using your character? Still image, game character, animation, etc...
• Will it speak, make noise, bare it's teeth? (i.e. will it need a full mouth setup or can you model it closed?)

Once you have decided on what you want your character to look like, it is time to get reference material. If you draw well, you can sketch out front and side views of your character to load into blender for reference. If you don't draw well, you should look for images that closely resemble what you are going for. The internet is a great resource for reference images. Another option, (one I use a lot), is taking a trip to my local department store and heading straight for the toy aisle. Children's toys make great reference objects for modeling. They are highly detailed and come in every type imaginable.

You have your reference images loaded into blender, now what? Well, now you start modeling. There are various techniques available to complete your model. Choose the one that best fits your workflow and gives the best results for your model. I personally prefer box modeling, but I have been known to use a variety of methods to get my model to look just how I want.

When modeling your character, you will need to decide how much detail you want to model vs. how much of it can be added later with textures. Even if you decide that a lot of the details can be added later, you will still need to make sure that you have added enough detail for proper posing/animation. The most common areas that need added detail are the joint areas. If you don't add enough detail, the joints won't deform properly and your model won't be seen in it's best light.

This wraps up our run-down of things to consider before starting your character. Now fire up Blender and get modeling! Your characters are waiting to make their big debut! ■

Project Orange

The long awaited "**Elephant's Dream**" was premiered March 24th at Cinema Ketelhuis, Amsterdam. By all accounts, it was a resounding success. This film marks an important achievement in Blender and open source history.

After the premier and workshops held that weekend at Montevideo (Keizersgracht 264, Amsterdam), the Project Orange team finished up last minute work and headed for home. But the project still had a lot to accomplish, namely, the production of the promised DVD.

Artwork for the DVD disc and package sleeve has been finalized and Joeri has been hard at work getting everything ready for DVD production. As of April 24th, the masters have been finished and all files sent to the reproduction company.

The DVD will include not only the film, but all project files and a documentary for us to drool over and take apart for our own education. By time this article comes out, most of us should have received our purchased DVDs and those who didn't purchase it can look forward to downloading the files shortly thereafter.

A huge round of applause is due for the Orange team and all they have accomplished. Their hard work has been an inspiration for us all ■

Google summer of code 2006

Once again, Blender has been invited to take part in the Google Summer of Code. Google Summer of Code is a program sponsored by Google for students to apply for a summer coding grant.

Taken from a news item posted by Ton:
Last year over 400 student projects were granted (10 for Blender), and Google expects to approve this year even more.

Although we've created a list of ideas for projects, we especially invite students to submit projects based on past experience and competences, or based on their current research, so that they can efficiently bring in new development directions for Blender. Coordination of the Blender SoC projects is again done by Chris Want.

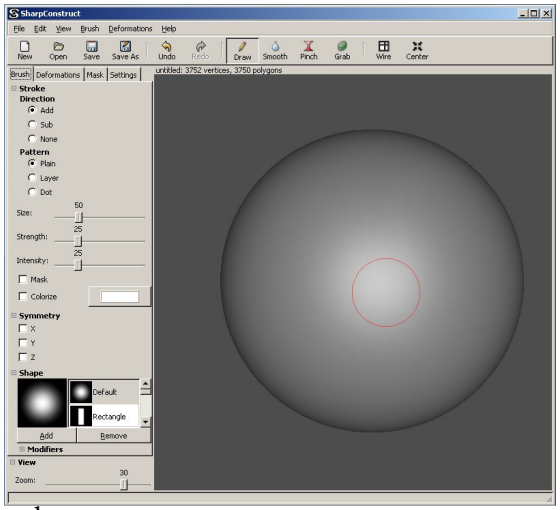
Students interested to work on a Blender project during July/August can find the information in the links below. Project applications will start on May 1 and will end May 8 by 17:00 Pacific Daylight Time ■

Blender SoC information:
http://mediawiki.blender.org/index.php/BlenderDev/SOC_2006_ideas

Google SoC information:
http://code.google.com/summerofcode.html

Sharp Construct 0.12

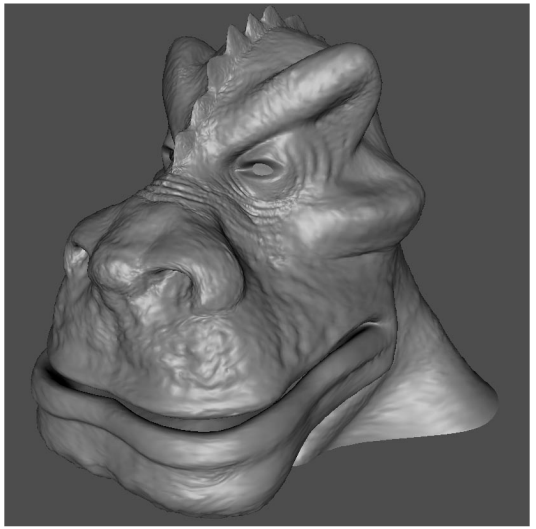
SharpConstruct is an 3D modeling application that allows you to work out on details of your previously created 3d model. You can even create the models entirely in it as well. It allows you to work on an 3d model as if you are working on clay pushing and pulling the surface into place to create shapes. The variations in brushes increase the level of operatibility for the user. On the fly working on Sub-div surfaces makes it an excellent tool, the responsiveness levels at decent enough dense meshes makes working in it quite pleasurable.



SharpConstruct.

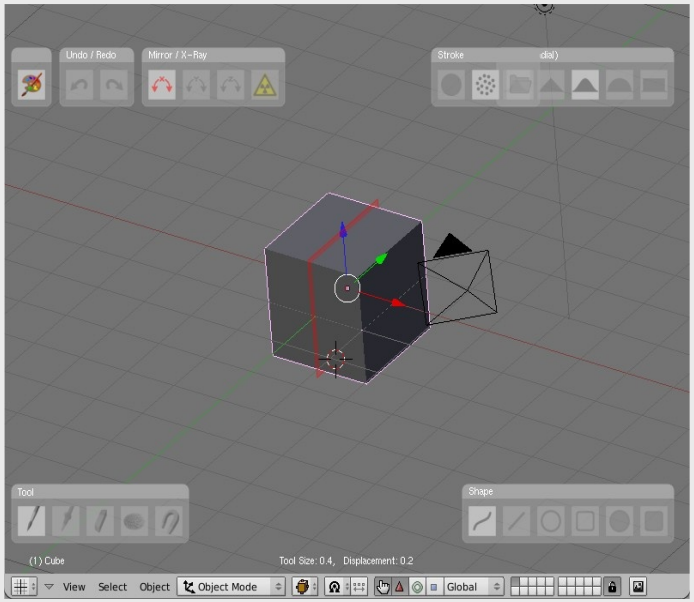
Sharp construct have been in news lately owing to its speedy development, it has become much stable and usable with the release of version .12x. Due to its nice enough interoperability between Blender it make for an pretty useful tool for Blender-heads, although Blender have it share of tools that mimic functionality of SharpConstruct but none is more feature rich than SharpConstruct comes in the genre of Zbrush. Although it is no way as advance as Zbrush itself but given that it is very young project and has shown a lot of potential up till now, we can expect it to become one of the most powerful tools Open-source has to offer.

It is available for both Windows and Linux at http://sharp3d.sourceforge.net



Expresso 1.1.1 Release

Blender has features that are similar to SharpConstruct, and in the form of two scripts. One comes with Blender 2.41 and another more mature script, called Expresso. It is an e xcellent python script written by Michael Schardt. It works in real-time within Blender.

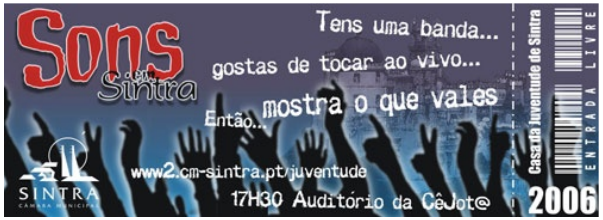


Expresso1.1.1 interface

It's interface is presented neatly within Blender's 3d window. Expresso has the most intuitively designed python script for Blender, ever.

The current version1.1.1 worked with out any problems in Blender 2.41 for Windows but on Fedora Core 5, it crashed even before it started, a constant annoyance with python scripts ■

You can find it here at http://members.fortunecity.de/pytablet/



Festival de Video- Portugal
SECOND SINTRA VIDEO FESTIVAL

After the success of the first edition, the Sintra City Council and the "Let's Make Television" programme, are once again organizing the Sintra Video Festival.

"Advertising" is the theme of this year's festival, which is aimed at all young people under 30, residing in any part of the country.

Works may be submitted in the form of spot advertising, documentary, or video.
The prize - giving ceremony will take place on April 22, and will distinguish the best film, the best idea, the best film of Sintra area, the best actor/actress, and the best director.

Rules and applications are available from www2.cm-sintra.pt

Please note that the web site is only in Portuguese Language ■

Character Modeling at Plumiferos

by
Manuel “Picasus” Perez and
Claudio “malefico” Andaur

Modeling a character is always challenging because it's always difficult to get a good topology in order to get nice deformations and expressions. For our film project “Plumiferos”, we had to model several characters among humans, cats, different types of birds and even a bat! Since they have completely different shapes and anatomies, we had to study each case separately.

At Manos Digitales Animation Studio, we work our models starting from a very solid character analysis made in collaboration with the Art Department. Designers have long-hour talkings with the Director until a look that reflects the character's personality is achieved. However, sometimes a look that is appealing for designers will not work in 3D or it will take too much time to skin/deform correctly. In these cases, we suggest modifications according to our own experience.

Animal Planet

The Art Department not only made drafts of the views needed to model the character, they also made drafts of the character's expressions and movements. It's very important to pay attention to these sketches since they will be the keystone for mesh topology.
When this material gets to us modelers, we draw all edgeloops on top of them. Then we use different modeling techniques. Some of us prefer the face-to-face (or poly2poly) approach where we create the faces of the main edgeloops first, and then join them.

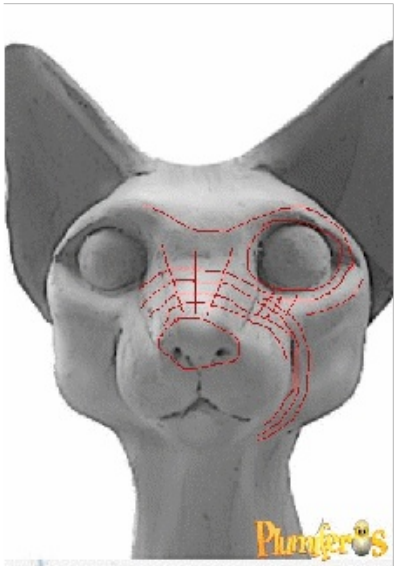


Illustration 1: Edge-loops drawn over clay model photograph

Illustration 2: Edge-loops drawn over pencil sketch

And, some of us use subdivision modeling (or box modeling) which is starting from a cube and subdividing and conforming the mesh. In any case, we start our work as if the characters were completely symmetrical.

This speeds up the process. We model only one half using a "Mirror Modifier" to get the whole picture as most people do. When the model is finished, these modifiers are "baked" and non-symmetrical details are added.

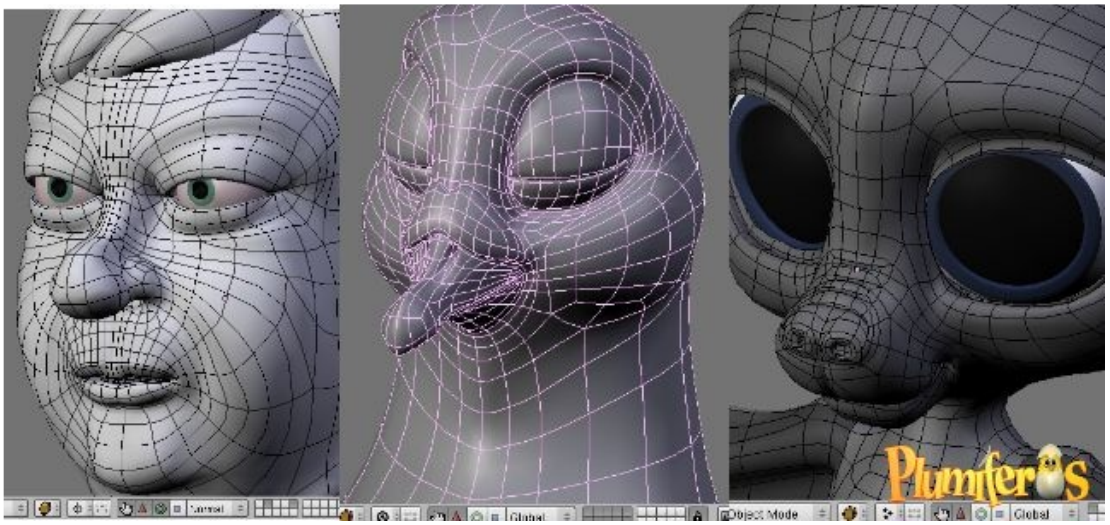


Illustration 3: Three characters with different anatomy: Human, Pigeon, Bat. Edgeloops are pretty similar in the end.

One of the first problems we had was to figure out how to deal with the bird character's mouth, since birds have no lips nor nose. After some experiments, it was clear that the bird's beak could be treated just as if it was a "combo" of mouth and nose. Creating edgeloops with this in mind made it easy to imagine. When beaks are too long, we change topology and keep the usual loops near the head.
Dental parts of non-bird characters are modeled as separate objects and then joined to the rest of the inner parts of the mouth.

Characters are made of parts

Characters are usually made of a unique continuous mesh. We decided to use multi-mesh characters for Plumiferos, meaning characters made out of different mesh objects working together, but never joined. This approach has the following advantages:

* *We can have more than 16 materials per character.*
* *We can reuse some parts (weight painting and shapekeys included) for other characters.*
* *From an interface point-of-view, we can get different drawmodes for different parts of a character, speeding up the animation process without sacrificing detail.*
* *Most tools work better with less vertices or mesh sectors, like mirroring.*
* *It's more suitable for team work.*
* *We can change parts of the character for different scenes easily (ever tried to join two meshes with shapekeys?). eg: dress changes*

The only drawbacks of such a decision is the more complicated skinning workflow (you need to skin every part separately and then "stitch" the parts in some way). For all these drawbacks, Python came to rescue. There is a very handy python script, written by Campbell Barton, that we used a lot: Mesh Weight Copy.

Lord of the Wings

For our bird characters, we decided to model a “basic wing” mesh and from that, build the wings for all characters with only minor mesh editing. Since we are using multi-mesh characters, we could add wings at any time while animators are working on corporal or facial expression. For this basic wing to perfectly fit other characters, we had to design a proper “wing-body” interface that would work for all characters.

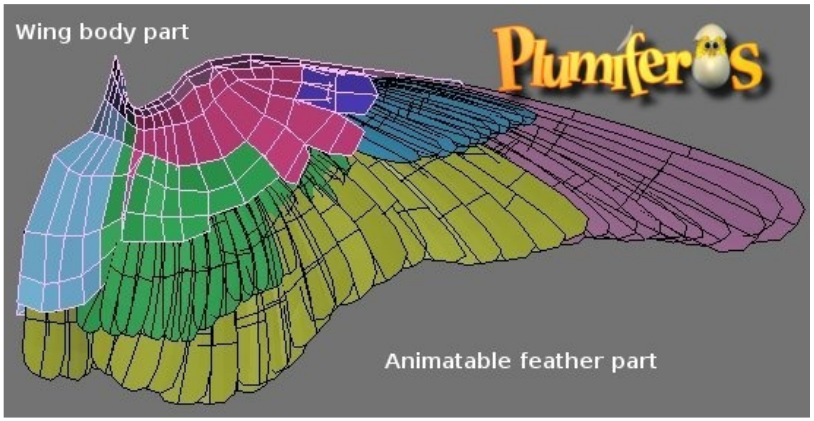


Illustration 5: "Basic wing" for all bird characters

When referring to feathers, I mean the main feathers that are used in the bird's flight, not the feathers covering the body. These feathers will be animated, for instance, when the character spreads its wings or shuts them.

The wing feathers pattern is a very complex and important detail.

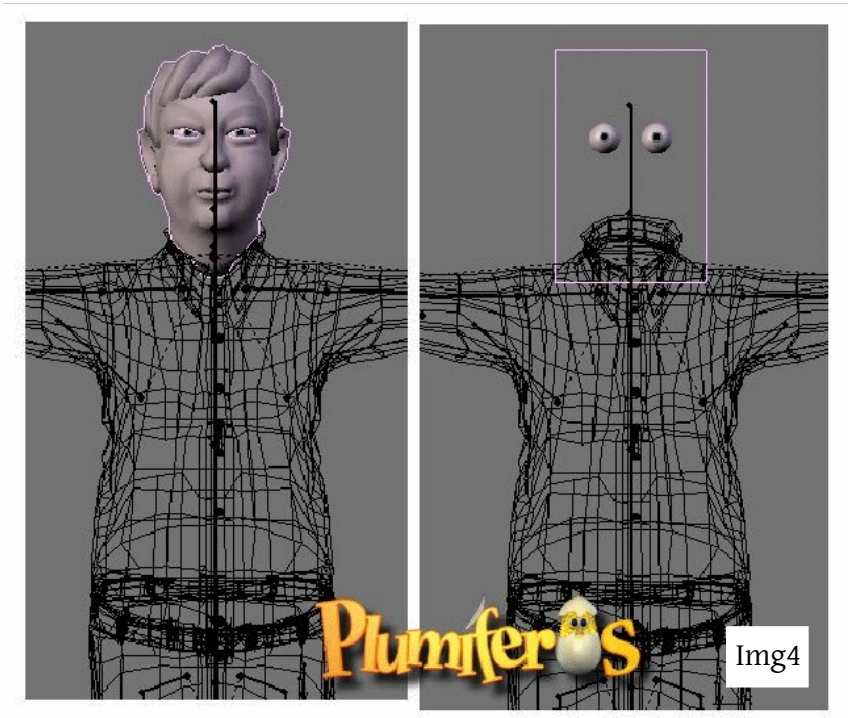


Illustration 4: Having multi-mesh characters allows weird drawmodes combinations

We used several references from all kind of sources including videos filmed by ourselves. Feathers were modeled and placed by hand as separated objects. Different plain colour materials were applied in order to better identify the types of feathers in the wing. All of these feathers were joined into one mesh.

Besides these “functional” feathers meshes, another mesh representing the main part of the wing was modeled. In this mesh, edgeloops were important since this part would act like a real wing, and would be deformed heavily.

The bat character required a completely different approach. Parts of the wing should behave completely elastic allowing the character to expand it or compress it almost like cloth, and other parts should look flexible like the rays of an umbrella. For the membrane parts, we tried to stay on the bare minimum vertex count, trying also to keep clean lines of deformation to avoid artifacts on wing bending.

Modeling and Rigging

Once the basic look and feel of a character is achieved, the mesh suffers further modifications in the skinning/rigging process. Here all topology problems, that weren't foreseen, show up in a sort of painful way. The worst part is that the mesh needs to be corrected without changing its external shape. So, as a reference, we used orthogonal renders of the original meshes.

If a mesh deforms poorly, it could be caused by poor rigging, bad skinning, or bad topology. Even when all these conditions are OK, sometimes a mesh just has a lack of edgeloops. Sometimes modelers, seeking a minimal amount of vertices, forget to add enough edgeloops to guarantee proper deformations.

We test the meshes in different actions and see how they behave. In this stage of the process, we used to model cloth wrinkles, or change edgeloops directions until we got the nice deformation we wanted. For this kind of job, we edit the mesh in its “rest” state, or if needed, use the armature's deformated state (the so-called “Crazy Space”).

Illustration6: Basic wing applied to main character

Final Words

I wish we had more spare time to write a more detailed article but, our production schedule is very tight. :(

We would like to say Thank You to all the Blenderheads around the globe who are supporting our project and helping us in many ways, to Ton and the Orange Team for their invaluable work, and to all coders who are making Blender bigger and bigger ■

http://www.plumiferos.com/

Modeling a naturalistic butterfly

by
Christian Guckelsberger
Level: Intermediate

Introduction

Because of the relatively high complexity of the modeling process, I decided to write this tutorial for the intermediate blender artists. There will be passages, where you need to continue working on yourself on the model until the next important step is being explained. At all, this is not a step-by-step instruction guide; it is an attempt to show you how to create organic models based on a butterfly with highlights on the most important parts of the animal. Do not forget that a modeling process is a process of continuous detailing and refining.

What is the model being made for?
I decided to make a film for a portfolio presentation, in which the butterfly plays the role of a guide. In that case, there will be many close-up and animation shots. Since the shots will be close, I made the butterfly relatively detailed, but not too detailed because of the long animation rendering times. You will need to trade off the detail of your model for yourself, watching it's final destination.

Preparations

I decided to model a naturalistic butterfly. Because I had not really had any clue about this sort of animal at all, I successfully rented a showcase and a book about butterflies from school. Trust me, the book did a good job after all, the butterflies in the showcase were too small to use them as a modeling reference. So before starting, you should have a reference, take a look into the library or the Internet. (Fig. 1 and 2) show the references that I used. There is a side perspective as well as a perspective of the half-front head.

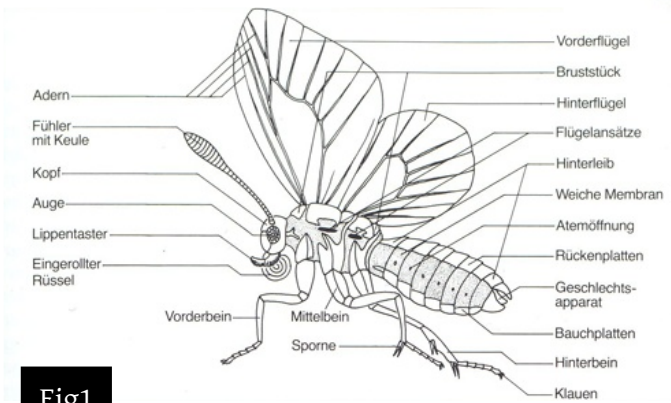


Fig1

Let's start. The abdomen.
As you should do with almost every organic model, first make a simple construction out of primitives and refine it later. I did not use a reference picture in the background, but you can do so if you like.

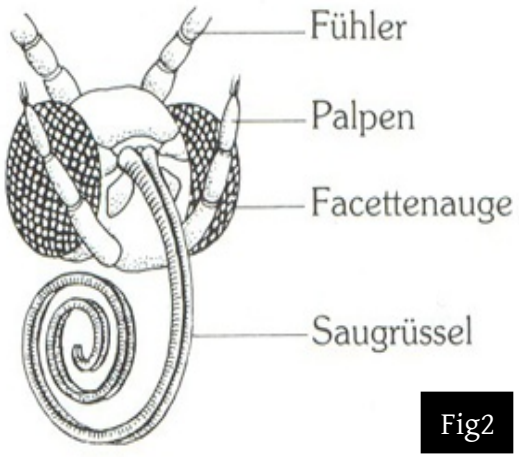


Fig2

Use View>>Background Image>>Image to add an image to your background.

Fine. Let's do the thing from the back on and start with the abdomen. I used a cube from the side perspective [Num 3] and extruded it 4 times. Then, I inserted two vertex rows by entering [K] in edit mode [TAB], using a loop cut. As an alternative, you can also use normal cuts [K] or multicuts [K] or subdivide it twice or several times, but with the last method, you will probably have to deal with unwanted vertice rows at the end. (Fig. 3) shows how the extruded cube looks after aligning it on the simple shape of the butterfly abdomen.

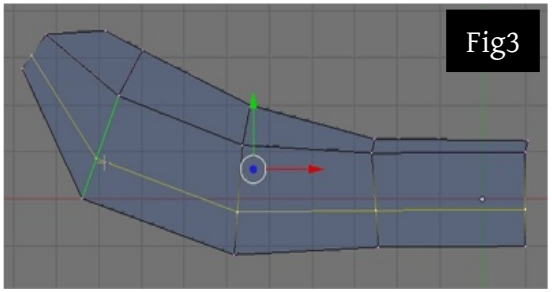


Fig3

Now you will need to continue extruding the mesh right to the thorax part (Fig.4). After aligning the primitive mesh to the butterfly shape from the side perspective, you will also have to do the same from the top one [Num 7]. Try to make the model smooth and round.

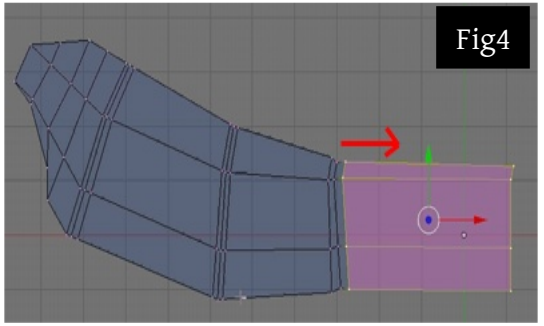


Fig4

You can also already use Subsurfs (Editing>>Modifiers>>Subsurf>>Levels) to see how the model looks smoothed. (Fig. 5) Pay attention to the highlighted vertex row in (Fig. 5). Later, it should become the border between the upper-body (insect shell) and the lower body. As you can see in (Fig.5) too, I also already added 2 cuts to every vertical one to indicate the later borders between the shell segments, as well as to

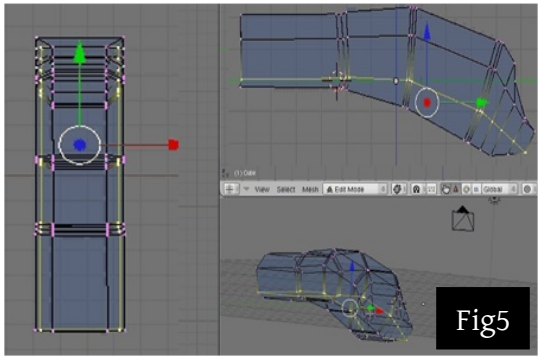


Fig5

refine the back-end of the insect.
Now it is time to define the different shell segments the first time. Select the middle of the 3 rows which indicate a shell division and scale it a bit smaller [s] (Fig. 6). You should repeat this with every row, then continue with the next step.

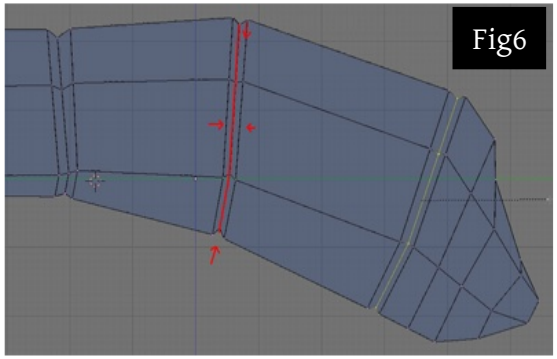


Fig6

Okay, now we will continue defining the different segments of the abdomen. To define the upper shell, use the face-select mode and select all faces that are lower than the border line between the shell and the lower-body (Fig. 7).

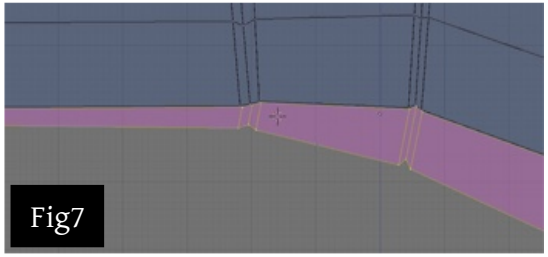


Fig7

Now we use [E] to extrude the lower part and insert one more row of faces and the scale-tool [S] to scale the whole lower part just a bit smaller. Let's move the whole thing [G+MMB] vertically until both lines (the upper line of the selected segment and the shell border line) nearly cover each other (Fig. 8).

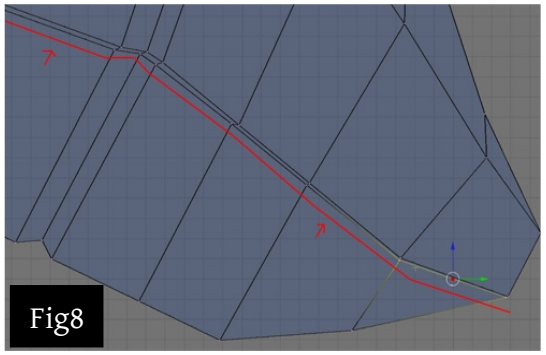


Fig8

As you can see, the vertices of the upper row of the selected segment do not stand directly under the ones of the border line anymore. Use the scale-tool [S+MMB] again and just scale the lower part horizontally this time, until each vertice appears under its neighbor again. You can also use the move-tool [G] for this, but it definitely will need more time because you will need to move every pair of vertices on their own.

The Thorax + Head

Proceed extruding the mesh right to the head and align it to the insect shape again. Do not forget to align it also from the top perspective! :) For the head, we can use a simple box subdivided by the lines we already got, so just extrude the thorax again a little bit. Look on (Fig. 9) to see the whole process so far. You

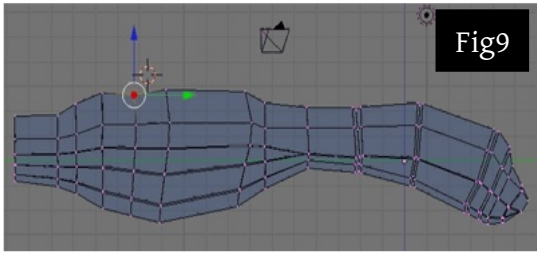


Fig9

should also pay attention to the refinement of the back-end which I already did. Now continue to make the head smoother, making it look like a low-detailed half-sphere from the side perspective. Use the move-tools [G] to achieve this by easily moving the vertices which we already got due to our former cuts (Fig. 10). (Fig. 11) shows an image of the model we got so far with Subsurface turned on (level 2).

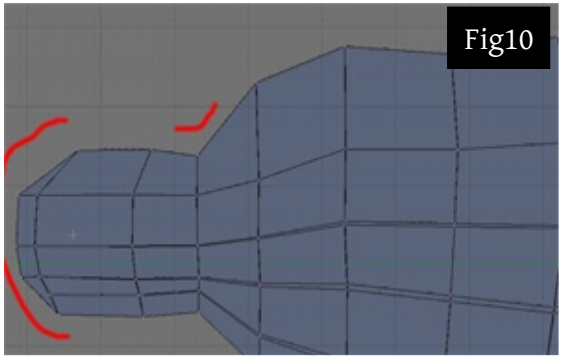


Fig10

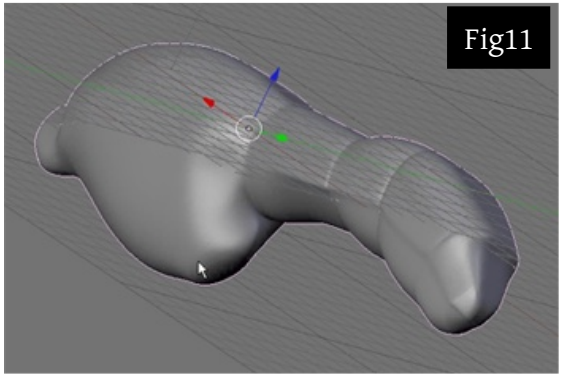


Fig11

Details of the head

As you can see, the butterfly on the reference picture has got antennas on his head. There are different, breed-specific antennas but we will try to imitate these ones. Select two faces which are large enough to become the antennas on the upper head from the top view and extrude them by [E>>Individual Faces] the way it is shown in (Fig. 12). If you need to select two or more faces on each side of the head (each side should be equal, as mirrored if you did everything right so far), you need to extrude the antennas separately.

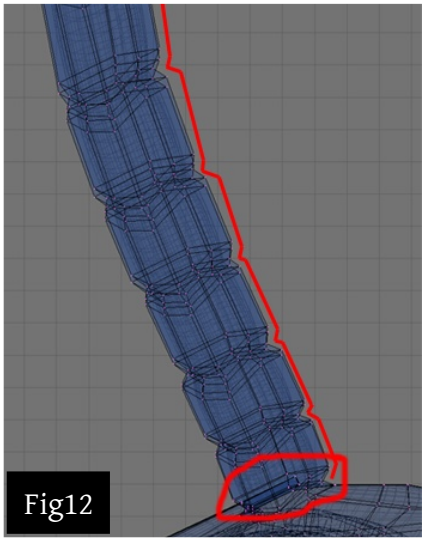


Fig12

If you need so, do it with one, then copy it and connect the lower faces with the faces of the head [V] as you can see in (Fig. 12). I first did the extrusion with one or two faces, then inserted some more rows in it and scaled them a bit vertically/horizontally from the top view to make the antennas a bit smoother, not that cubic. Try to give these parts of the body their specific look by extruding straight, extruding and sizing, extruding and sizing again and then extruding straight again... also try to make the individual segments different from each other to give the model a random look.

As you can see on the reference image, the antennas become larger at the top, so

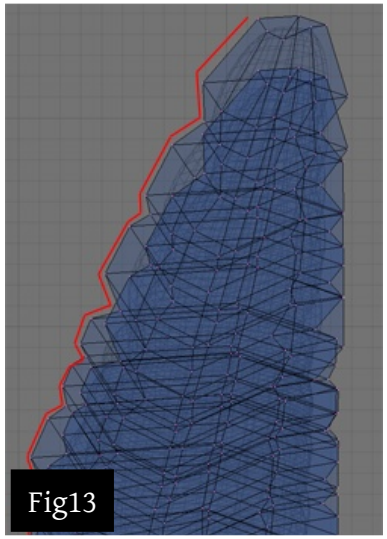


Fig13

let's do them as shown in (Fig. 13). You should also always take a look on the other perspectives (front [Num 1], side [Num 3]) to prevent an unwanted shifting of the vertices. In both last Fig., I already bent the antennas a bit to the left and right (seen from the front) by selecting some of the upper vertices, turning Proportional Edit Falloff on [O], selecting a given falloff method (here I used a combination of smooth and root falloff) and moving the


Fig14

whole thing with the move-tool [G]. Probably you will need to experiment a bit with the right falloff method to reach satisfying results. Try it on your own. If there are also unwanted vertices, e.g. from the neighbor antenna shifted with, try to move the other antenna far away with Proportional Edit Falloff off and shift them back later (Fig. 14 shows the whole, finished antenna of my model).

Fine. Let's make the head a bit more complex now. I added facet-eyes to the head by adding an Icosphere (4 Subdivisions), moving it to the correct place from the front and side perspective and erasing the part of it that protrudes into the insects head. I used to create the Icospheres out of the edit mode [TAB] to have them as a single mesh and on that way, separately for the later texturing process. Finally I got a bit more than a half Icosphere on each side which I fitted on the head by using the Proportional Edit Falloff Tools [O] again. This action could need some time until the whole eye fits (nearly) perfectly to the head. So take some coffee and be patient. :) In addition, I also added a ring, made by extruding a circle several times (using the scale-tool [S]) and fitted it on the head as well, as a kind of frame for the insect eyes (Fig. 15). As a further effect, this reveals the unattractive area where the Icosphere hits the Head's Surface.


Fig15

After applying the eyes, we will continue with the second instrument these animals have on their head. As you can see on (Fig. 15), they start out as nearly as under the eye and long until the upper end of it. Their structure is relatively equal to the one of the antennas, so use the same technique to create them.

As you can see in (Fig. 16), I also created a beak which hoists spiral under the head. You can easily create this by extruding one face from the upper middle head and continuing extruding [E] and rotating [R] it until you have a spiral. I did the whole thing a bit more complex to give it a bit of structure as you can see in (Fig. 17). There is also a "mouth" (or better: hole) shown where the beak comes out. I took this screenshot with Subsurfaces turned off, so you have a better insight into the modeling structure. As you can see above the beak (or mouth, seen from the top or front perspective), I had to make some more cuts to define them. Do not use loop cuts [K] all the time. Instead of this, use normal cuts [K] to increase just the part of geometry you really want to increase in detail and avoid problems with too many faces or problems with too chiseled edges (This occurs when too many edges come too close together on other regions of the mesh, e.g.) when turning Subsurf on again.

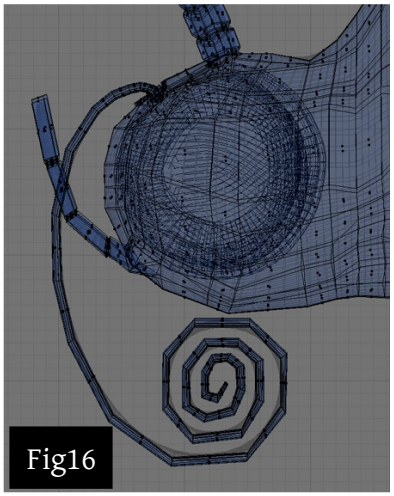


Fig16

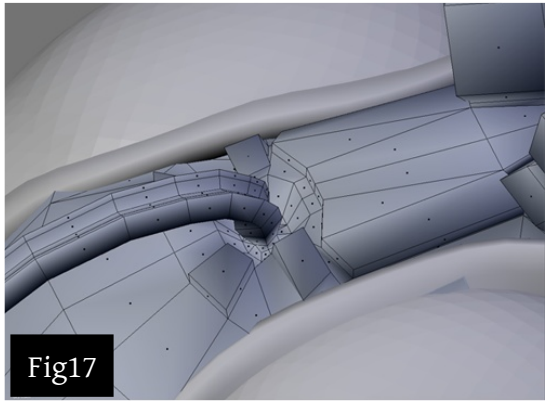


Fig17

The legs and their connection to the body.
As seen on the reference image, the butterfly has a shell around it's legs as well as on the part which finally connects to the rest of the body. This part looks relatively complex which would need a lot of additional vertices in the thorax geometry and on that way could probably disturb the Subsurf functionality or easily increase the vertex count unnecessarily. Because of this, I did these elements as well as the legs separately. So - let's get of edit-mode [TAB] and first create two legs as seen on (Fig. 18). I started out with a circle (about 5-7 vertices) from the top view and extruded it until I reached a model as shown in the figure. To create the individual shell segments, you can use the same technique as already done with the abdomen or the antennas. To make the individual shell parts shifting under each other, move [G] and scale [S] them until they reach a bit under the neighbor shell segment, as seen in (Fig. 19).

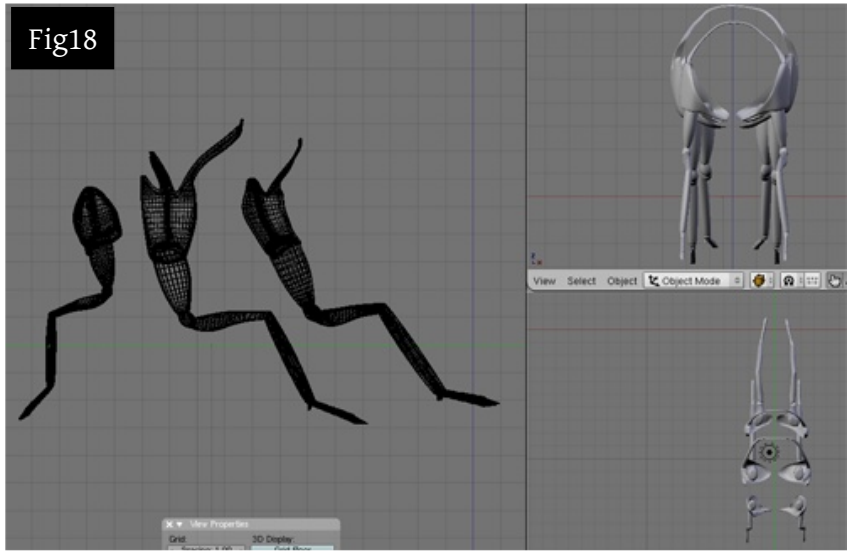


Fig18

There is also the lowest part of the insect, its claws, shown. Modeling them highly detailed increases the visual complexity of your final model. We will use the technique shown here to refine the abdomen later in the details section. Now we should have models as in (Fig. 20) and in (Fig. 18, the other legs).

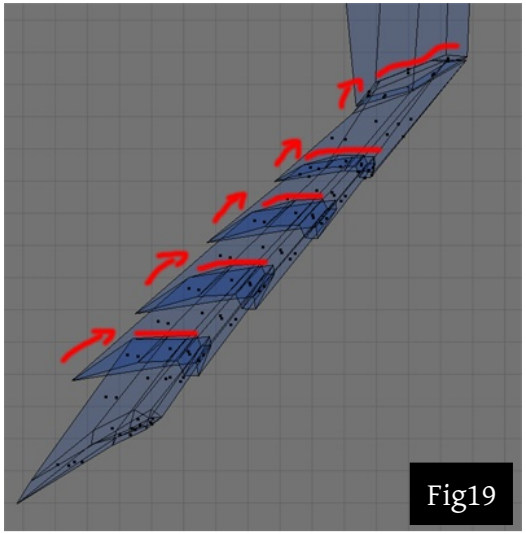


Fig19

Now we will start out with the connections to the body. I did a cube at first, flattened it a bit [S+MMB], erased the bottom face and then inserted some loop cuts [K].

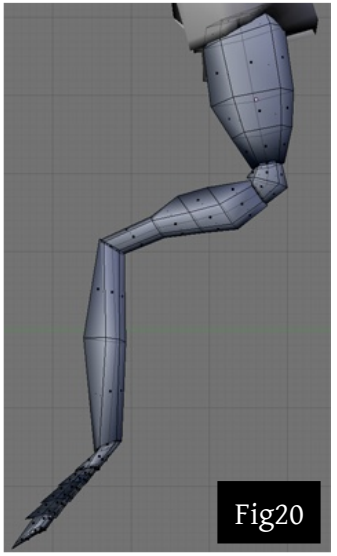
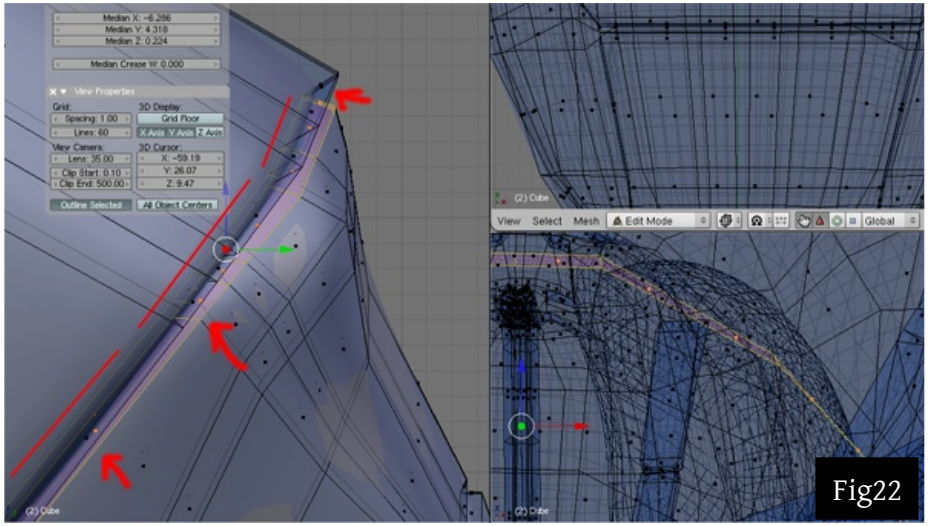
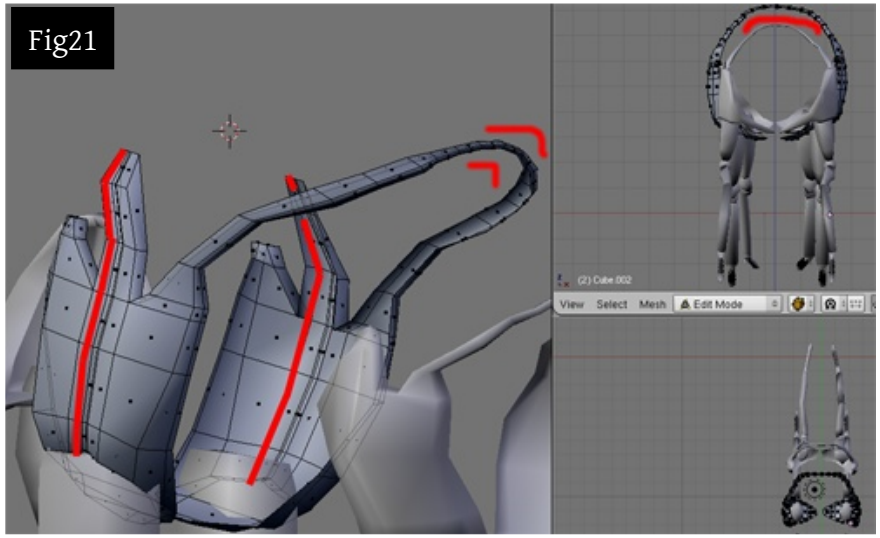


Fig20

I continued moving the vertices until I reached a form as in the reference images. Then, I fitted them on the thorax by using the Proportional Edit Falloff tools from the 3d- and side-perspective [Num 3] and extruded one edge until they reached the upper middle of the thorax. As a last step, I mirrored the part and joined the two meshes by hitting [J]. Now I connected the parts I extruded before to the upper middle of the thorax together (Use [V]-key while having 2, 3 or 4 unconnected vertices selected to create a line, a tri, or a quad). That's it. Now you just need to make 2 variations of this form for the other 2 pairs of legs. You can also create some deep-structure by making 3 vertical cuts again and moving the middle of them along the z-axis of the model. You can see this in (Fig. 21). Look for (Fig. 18) to see the connection parts from the side perspective.





Refining the main body

Okay... the model isn't looking bad, but there are still things that need to be more intensive, expressive to look naturalistic. For example: The shell segments (again! ;)). We will use the same method of moving one shell segment under its neighbor as we already did with the claws. Select one row of faces in face-select-mode, scale them a bit smaller [S] and move [G] them under the other face row. (Fig. 22)

Now we will finish-up the back-part of the insect. As seen in the reference again, there is a kind of incision in the shell where another part of the body is visible between the upper and lower shell. You may also model this the following way: First elongate the upper part a bit and delete the faces between the upper and lower part. After this, extrude the border faces of the open lower part to the middle and secondly to the innermost part of the insect. Then, I extruded it again out of the mesh and formed the body part which is marked in (Fig. 23). Finally, do not forget to close the open mesh by selecting 4 vertices and hitting [V].

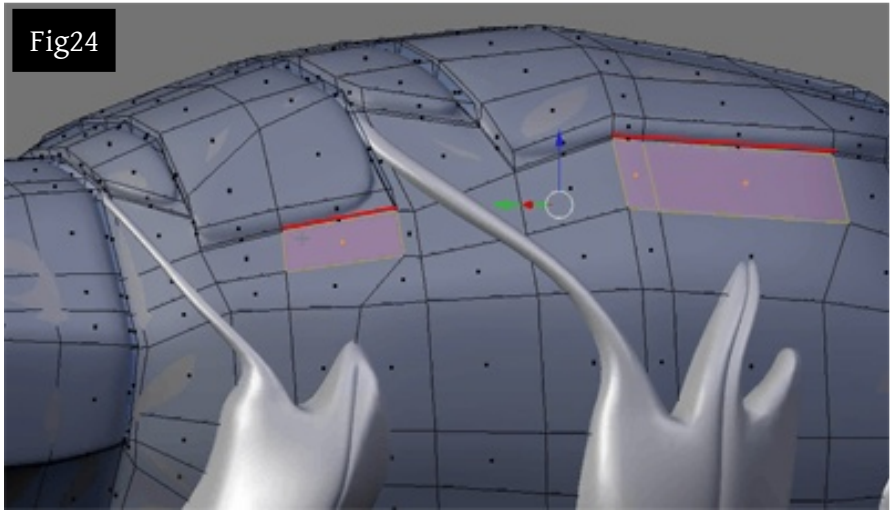


Fig24

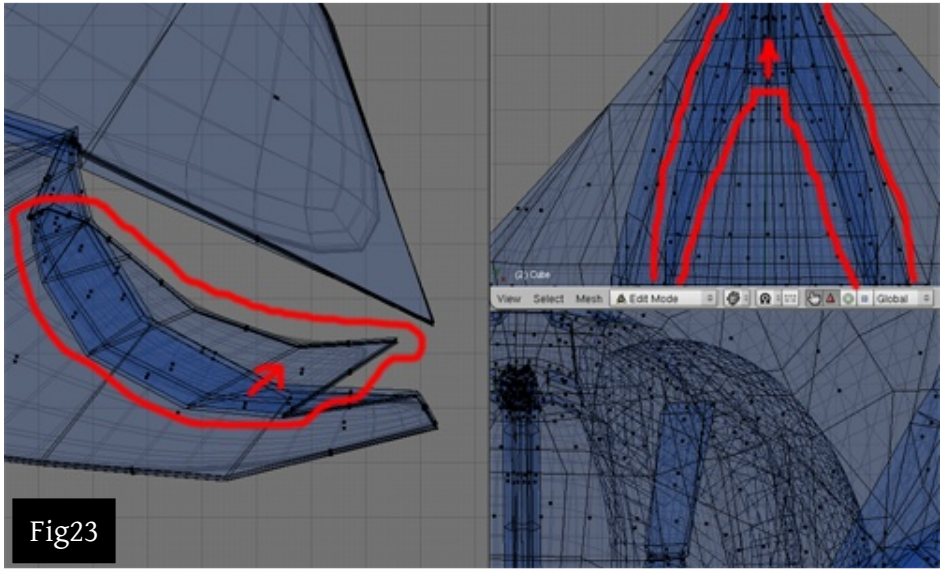


Fig23

As a third detail, you can extrude some chosen faces of the top thorax and extrude them too, to create also some shell segments on this place. See (Fig. 24) for this

The wings

As you can also see in (Fig. 24 and Fig. 25), I selected 2 pairs of faces and extruded them on both sides of the body in the top perspective [Num 7]. You can use the scale-tool [S] in that case to extrude them away from the midpoint after hitting [E] for extrude. Now, start to build a construct of muscles and/or veins which start at the body and become smaller and smaller in the wings region. The look of these muscles/veins differs from each butterfly too, so take a look at your personal reference or easily do it as you like :). Do not forget to make the segments a bit thinner at the end, using the scale-tool [S] again. Please do this whole action for just one side at first. Now select your one-side construction, mirror it, and attach it to the other extruded ends on the other side by [V]-key (Fig. 26).
Okay.

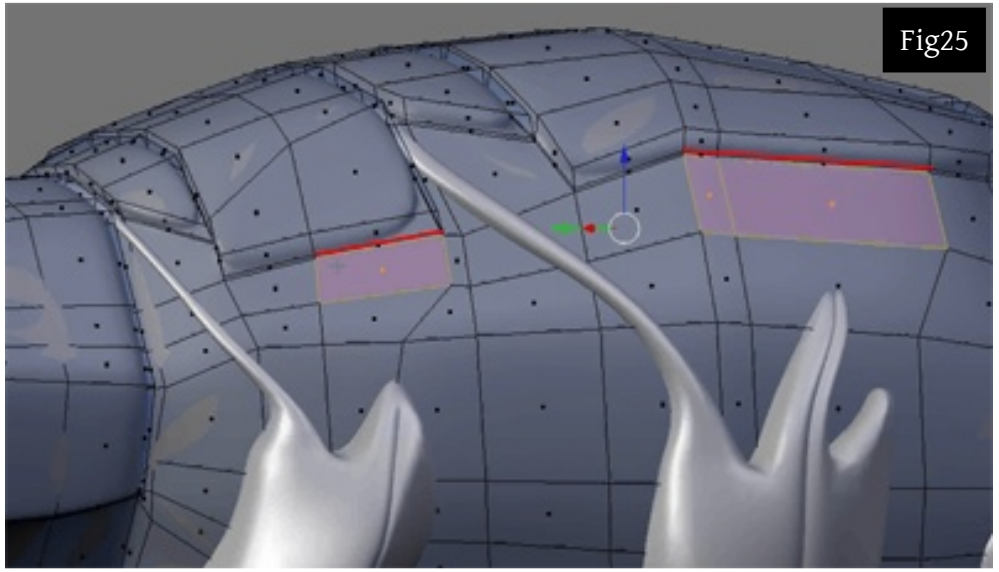


Fig25

Let's start with the wings surfaces. There are several methods to achieve this, e.g. 3d splines, polygonal surfaces... etc. For now, we will continue with our technique we used so far: Go out of edit-mode [TAB] to make it easier to texture later. Now start with a simple cube again and flatten it so it becomes very thin (but don't forget that it probably becomes even thinner when Subsurfs are turned on). Try to create two shapes (Fig. 27) by extrusion [E] and refining ([K] for loop cuts or normal cuts etc.) whereas the upper one (as seen in (Fig. 28)) overlaps the lower one just a little bit.

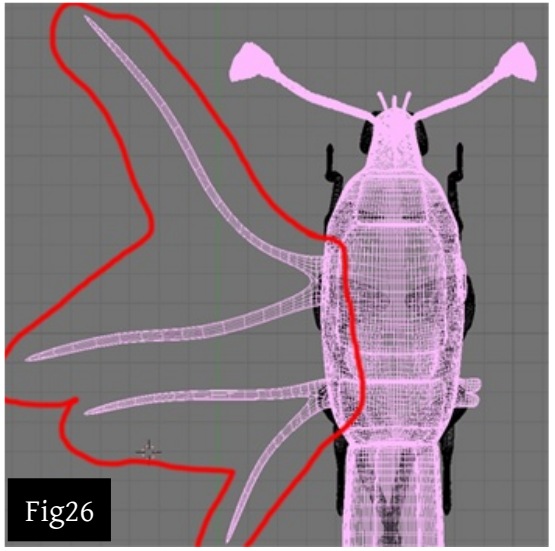


Fig26

Now, turn subsurfaces on and make the wings in depth a bit irregular. For this, we can also use the Proportional Edit Falloff-Tool [O] despite we could also use lattices, etc. But for now, we will try to minimize the assortment of tools used and take the first one. You should pay some attention to the wings that still overlap, but nearly connect to each other with their lower and upper surface. In (Fig. 29), the depths of the surface I used here are marked, just for example.

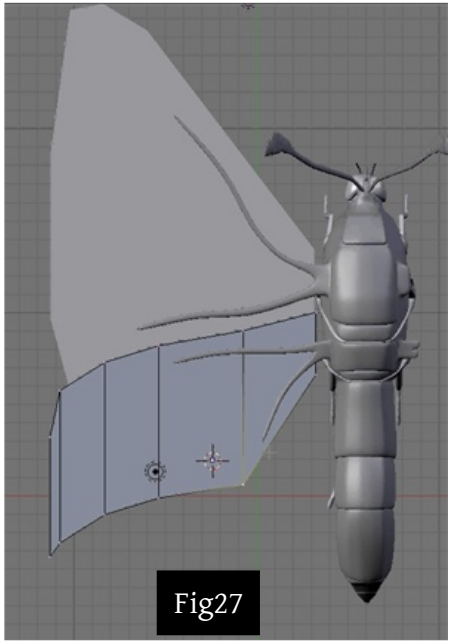


Fig27

Now comes the last step we need to do: We will need to take a look to see if the muscles or veins we did just some steps before are still visibly connected to the wings in top [Num 7] and bottom perspective. If they are not, try to achieve this by moving or welding them with the Proportional Editing Tools [O] and/or the move-tool [G].

Okay - now recline and behold your model - did you ever imagine that a butterfly looks like that? Check your reference - if it's identical, you did a very good job.

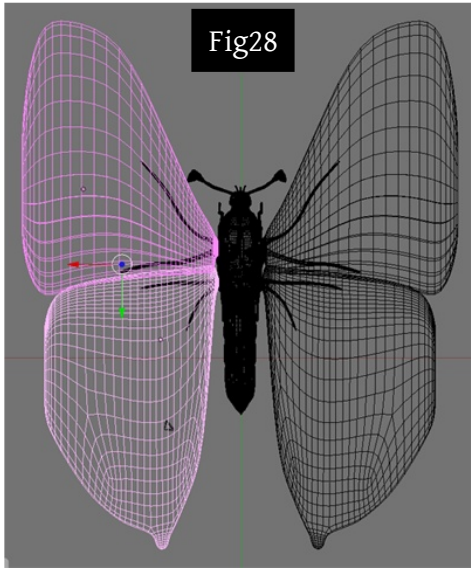


Fig28

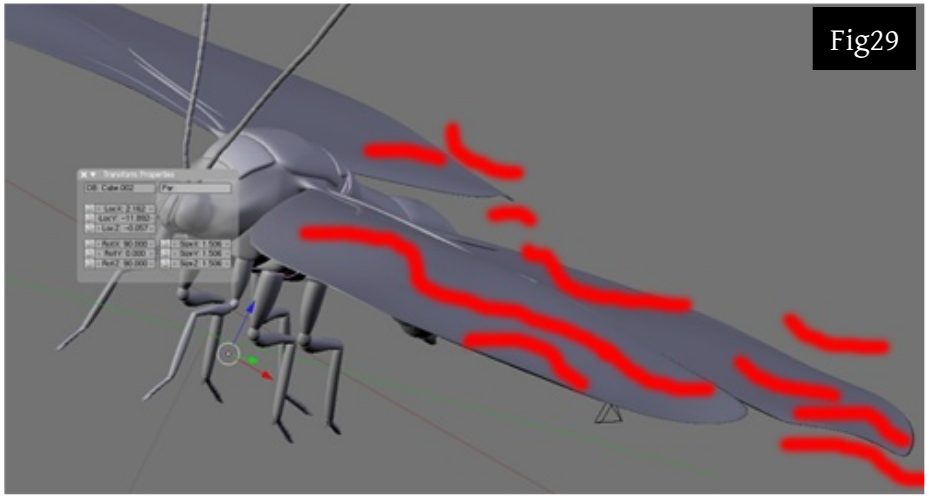
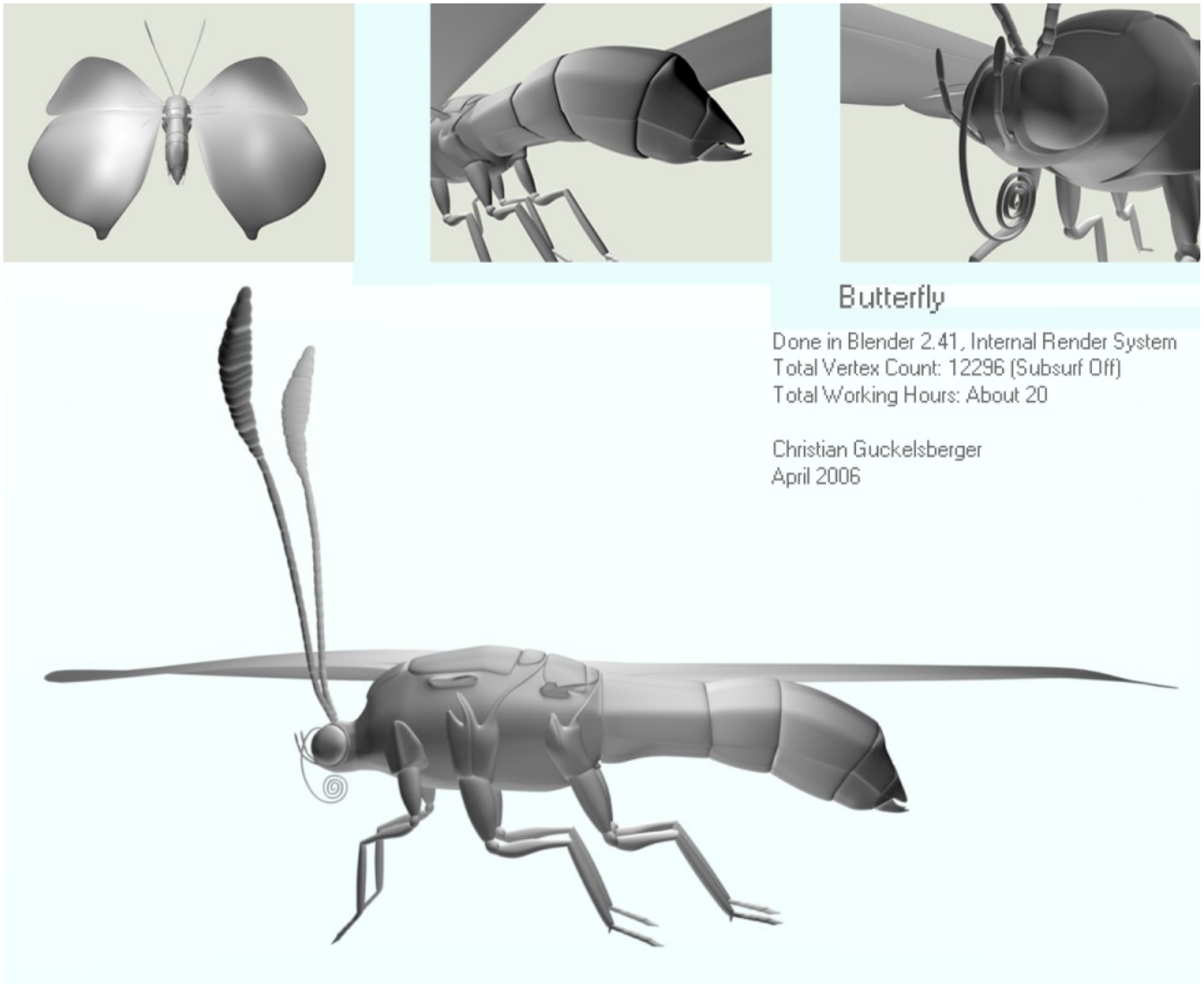


Fig29

The final, yet untextured and unrigged result can be seen below. I hope I have successfully covered all aspects of the model and have written a comprehensible tutorial ■



Rapid Prototyping with Blender for Jewelry/Metal Artists

By

Claas Eicke Kuhnen

The Making of Special PEZ Dispensers

In my recent MFA Thesis Exhibition, I presented PEZ dispensers which were constructed with found objects, hand fabricated and computer generated elements. In this article I will focus on the technical possibilities that Blender provides artists in creating CAD generated elements.

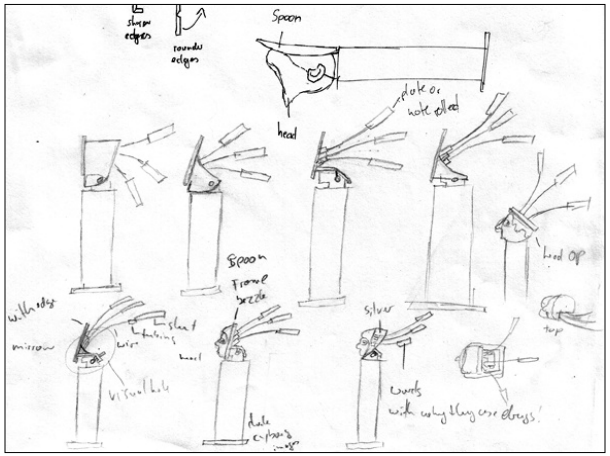
Together, Blender and Yafray are a very strong toolset which provide all needed modeling and rendering tools to visualize ideas and explore variations in a short period of time. With it's excellent modeling and animation tools, Blender allows the designer to virtually create his object and proof mechanical elements. Yafray takes core for a photorealistic rendering.

However, in terms of modeling, Blender also offers the possibility to export 3D geometry into the STL (Stereo Lithography) format to transcribe to 3D printers.

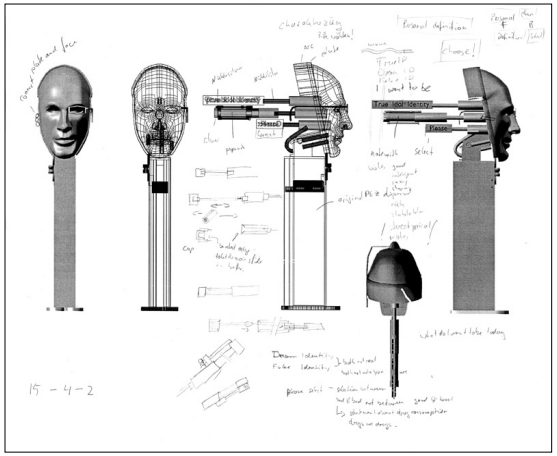
It is important to keep one limitation in mind about Blender: Blender does not support advanced NURBS tools. And NURBS tools are essential when designing shapes and elements that require a high amount of precision. And polygons do not offer that precision either. However, when a NURBS object is exported into the STL format, the surface is tessellated and saved as a polygon mesh. Good results will require a high tessellation of the NURBS objects. This is similar to working with Polygons. While polygons cannot replace NURBS, they have their own advantages. Organic modeling is today's strength of Polygons. Together with Subdivided Surfaces, you not only have the easy modeling tools to quickly create your desired shapes, but also Subdivided Surfaces that are needed for fine tessellation to gain smooth STL files.

During the past three semesters at Bowling Green State University, I have explored a design process in which I include traditional and computer-aided approaches to explore, develop, and construct my art objects.

This process begins with few quick sketches to capture the most important ideas of the project. Hand sketching allows one to quickly explore many ideas in a very rough way. However sketches,

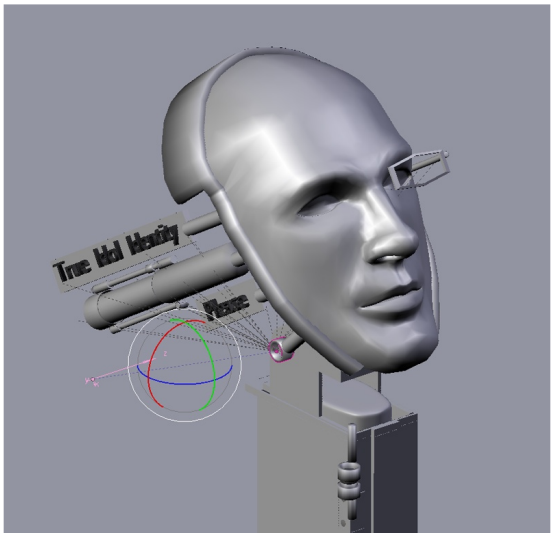


by nature, have a habit of being two-dimensional, static, and disproportionate. This is when Blender becomes an essential tool. After the main elements are worked out, Blender is used to construct ideas in a three-dimensional space and to refine the objects.



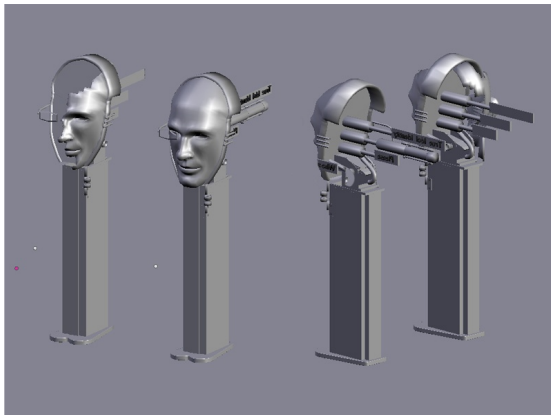
The advantage of switching to Blender after this beginning stage is that the main and important ideas are explored and a solid foundation is created on which Blender can rest. After an initial modeling session (which often requires quite some time), applying changes to the design, adding small elements, or changing the point of view only require few clicks of the mouse. It is during this stage that decisions concerning material thickness, scale and proportions are made. Specifically, when mechanical elements are involved in the design, their interaction will be explored and refined.

Working with the correct material thickness is very important when you are incorporating rapid prototyped objects into your art. For example, if it is too thin, the object might crack.



At this point, all of the important aspects of the design's development are complete. In the following step, orthographic views of the 3D model are printed out in 1:1 and used as a reference to start constructing the metal elements.

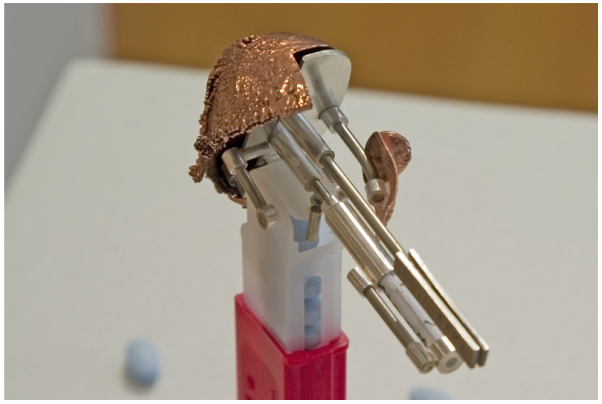
The majority of the metal work can be constructed using flat sheets of metal. The templates need to be sawed out, filed, bent, and/or soldered together. However, creating a realistic head out of a sheet of metal is not so easy. This task requires a different approach.



Thanks to Blender's STL export feature, the naturalistically modeled head could be sent to a zCore 3D printer to create multiple, real three-dimensional replications. During this process it was important to work with closed surfaces. The zCore software needs closed solids, or closed surfaces with a modeled thickness to calculate where it has to glue the starch powder together, which at the end will turn into the desired object.

The zCore 3D Printer works like a regular printer, but instead of using ink it uses glue. The printer starts by laying down a very thin layer of starch over which the printer head puts down glue drops where the 3D model requires it. After that the printer brushes another thin layer of starch powder over the first one and again glue droplets are applied. Through this process the zCore Printer creates your 3D replication layer by layer.

These fragile 3D prints should be sealed with a special and fast-drying epoxy glue to prevent any damage. This process works best when the 3D prints are placed in an oven to warm them up. The temperature will make the epoxy glue sink deep into the surface when applied. This prevents any glue remaining on top of the surface and covering any structural detail. Two coats of glue are enough to create a very solid plastic hull or surface.



The only remaining step is to Electroform the pieces. To electroform means that you are growing metal onto your piece. In this case, copper is used. To prevent any damage to the solid object or the electroformer, the starch powder inside the 3D print should be washed out when the 3D print only has a very thin epoxy glue sealing. It could happen, that some water will flow through the porous epoxy skin and make the starch expand when placed into the electroformer. This will result in cracked surfaces which, however, also can result into beautiful crack details when desired. After applying electrodag (a copper paint) to the head and sealing the inside with wax and attaching a copper wire to it, the 3D head can be placed into the electroformer.

For the final head, the left side of the face got three coatings of electrodag, while the right side only received one very thin layer. This resulted in the left side being strong and solid and the left side being thin, fragile and only partially electroplated. Through a longer plating process with a low amp setting, a solid and even copper surface was created. Towards the end of the process, the amps were increased to create a strong nodulation on the right side of the face. The nodulation turned into an effect, which looks like the head is decayed.

The final stage of the design process is reached. While most important elements were finalized with the first sketch session and later in Blender, during the process of assembling all elements, the design is tweaked and changed furthermore. The printed design plans are a perfect playground for some more refined sketching. Quite often is it the case that the problem solving requires you to go forward and backwards between these two approaches until the final result is reached ■

Rendering with Povray from Blender

By

José Mauricio and Rodas R. "Morfeus"

Although Povray is an excellent ray-tracing software, Blender users did not have many ways of using it cleanly until the release of Blend2Pov. Although Blend2Pov was born as a script, just like the JMS Blender-to-POV exporter, the script exported the XML generated by Blender for Yafray to a POV format. Now in their latest version (0.0.6a at the moment of writing this article), Ramon Carlos Ruiz (RC Ruiz, the script author) has successfully integrated it as the render UI within Blender (Figure 1). This provides almost seamless integration for PovRay, based on clever usage of the Yafray code. This article will teach you how to use Blend2Pov.

Installation

To make it work, you will need to install the Blend2Pov binary which you can find at Elysiun, and also the newest version of Povray. You may also want to install Megapov, as it is compatible to Povray and has additional features like HDRI support and Film Exposure.

Right now, there is no executable installation for Blend2pov, so you have to unzip the rar archive of blend2pov into a folder and rename the blender.exe to something like blender2pov.exe and copy it to the default Blender installation folder. You may also want to create a shortcut to it for your desktop.

Before going any further, execute the Blend2Pov program to make sure it loads correctly. If it gives some missing file errors, make sure that you placed the renamed Blend2Pov file in your Blender installation folder.

After the successful installation of Blend2Pov and Povray, you'll be able to access it directly from Blender's Render button just as you do with Yafray (Figure 1).

Features

Now we will describe each one of the buttons for the Blender2Pov GUI in the

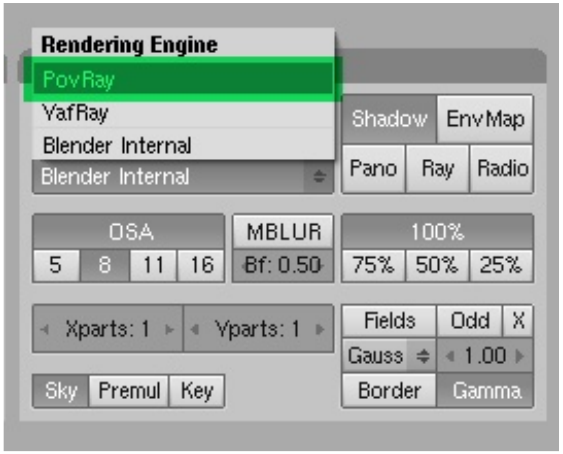


Fig1. Povray as rendering engine.

'Render buttons window'

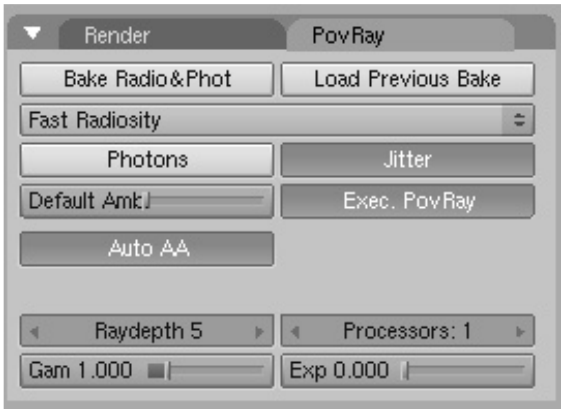


Fig2. The Povray GUI

Bake Radio&Phot

The purpose of this is to save the radiosity* and photons data in "exp.rad" and "exp.ph" files that are stored in the default Yafray path. If you want to render an animation with PovRay, it is recommended that you render the FIRST frame with this button activated. Then, disable 'Bake Radio&Phot' and enable 'Load Previous Bake'. Finally, render the animation. Doing this will save you some time on radiosity calculations.

Load Previous Bake

This button is used after you have saved the Radiosity and or Photon data in the first frame of your animation. It saves time during the rendering by using the saved data from the "exp.rad" and "exp.ph" files.

The quality of the radiosity is configured by default presets in Blen2Pov, which takes advantage of the stored values in the "rad_def.inc" files.

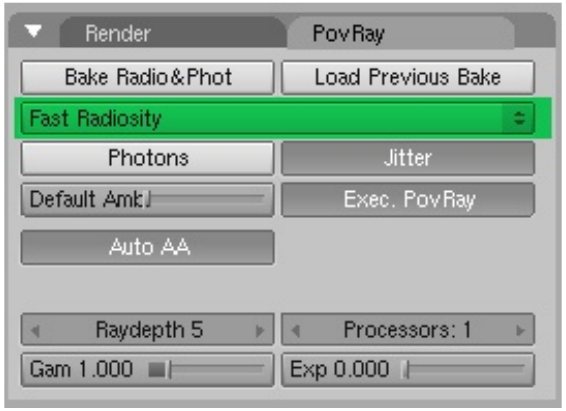


Fig3. Radiosity selector

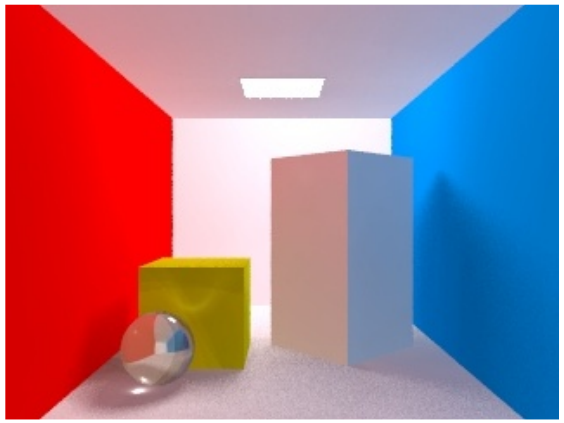
Fig4. Radiosity Options

Radiosity

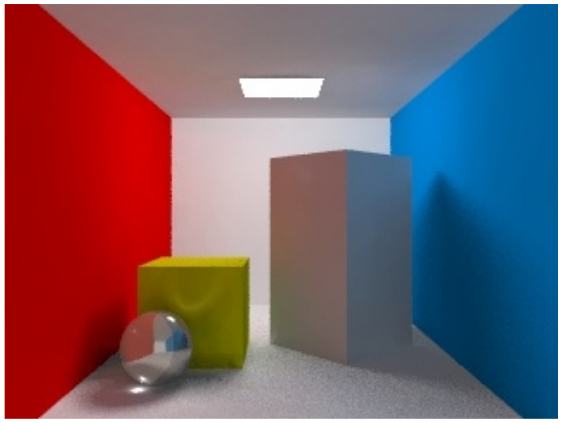
Radiosity is a technique used for the calculation of Global Illumination. Illumination is done by emitting light from each object in the scene. Objects absorb certain quantities of the energy and also reflect some part, for what can be considered as light emission by reflection. It is done in such a way that all the surfaces in the scene act as ambient lights, and therefore each one affects the illumination of the other surfaces (well-known phenomenon as Diffuse Inter-reflection of the light).

The calculation of Radiosity is based on mathematical methods that are dependent on Raytracers. Povray uses the Greg Ward Method for radiosity calculation. This provides a form of replacing the old calculations (constant value of ambience) with a value of light based on the interaction of light in the neighboring surfaces.

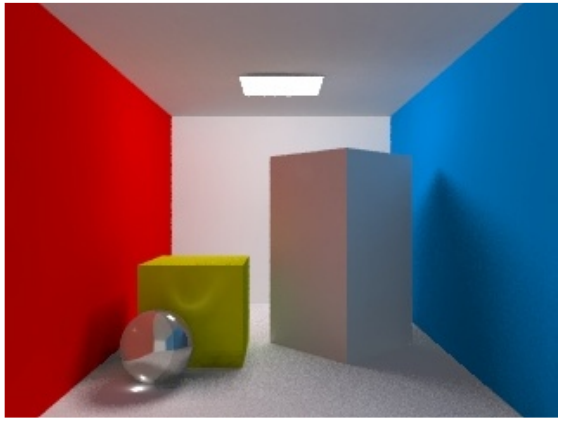
Now we will go through the available radiosity type options.
No Radiosity: When selecting this option Blend2Pov doesn't calculate radiosity.
Radiosity Default: As the name indicates it loads the default radiosity parameters, the results may be bad depending on the scene, but it is good enough to make test renders without sacrificing render time.
Radiosity Fast: This makes a quick calculation for radiosity, thus is of low quality but superior to the Default.
Radiosity Normal: This uses typical values of radiosity for scenes where generally a good render is required, but not sacrificing too much on rendering time (for example, animations).
Radiosity 2Bounce: This uses values of 'Radiosity Normal' but makes a double bounce of light, improving the scene. This is achieved by altering a special Povray parameter called 'the recursion_limit'. This parameter controls the quantity of times that the light interacts with an object. There is maximum of 20. In this case, the value of the recursion_limit is at 2. This also increases the render time.



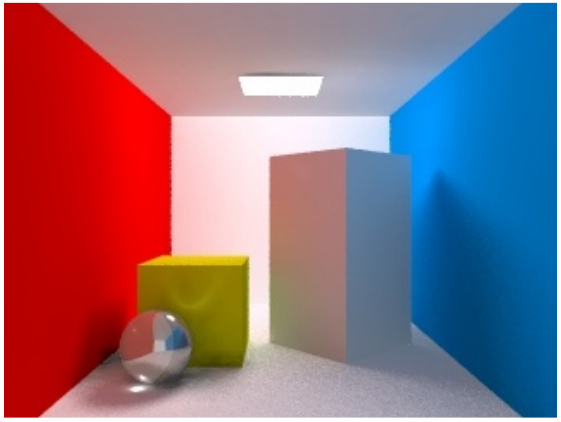
Default (20 Sec)



Fast (38 Sec)



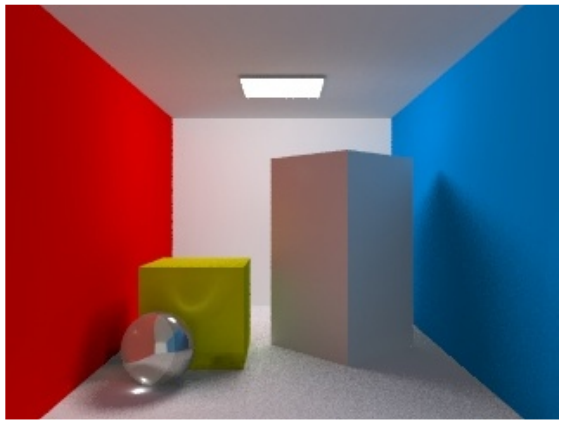
Normal (1 50 Sec)



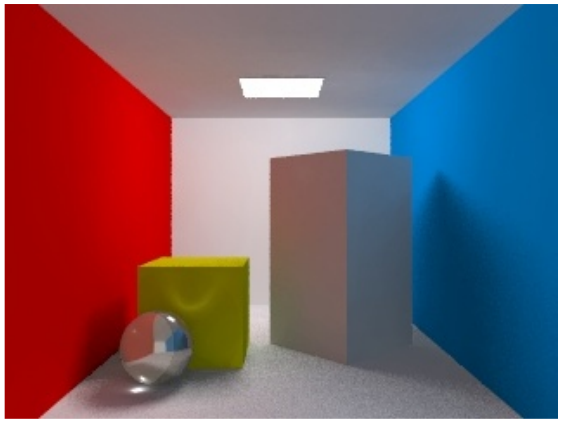
2Bounce (5Min 50 Sec)

Above comparison of the quality and render times with different radiosity on the same scene:

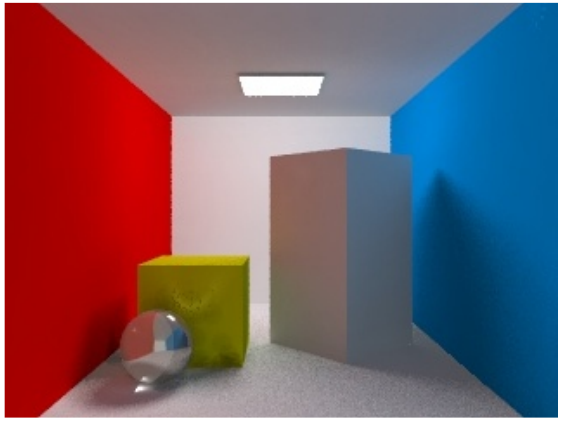
Radiosity Final: As the name indicates, it is used for high quality renders and as you guessed with even higher render times.
Radiosity OutdoorLQ: The low-quality radiosity rendering for outdoor scenes. It will be speedier. It is used to simulate radiosity in external scenes in which the light comes from a source of punctual light. The render is of low quality and is ideal for testing due to the low render times.
Radiosity OutdoorHQ: This is the opposite of the previous setting and is used for high-quality renderings of outdoor scenes.
Radiosity OutdoorLight: This is used to simulate radiosity in external scenes with the sun as a primary source of light. However, the quality is similar to OutdoorLQ.
Radiosity IndoorLQ: This is used to simulate radiosity in interior scenes. It uses a 'recursion limit' value of 2.
Radiosity IndoorHQ: This is used to simulate radiosity in interior scenes using a 'recursion_limit' value of 3. The render time is high so, this is best used for your final renders.



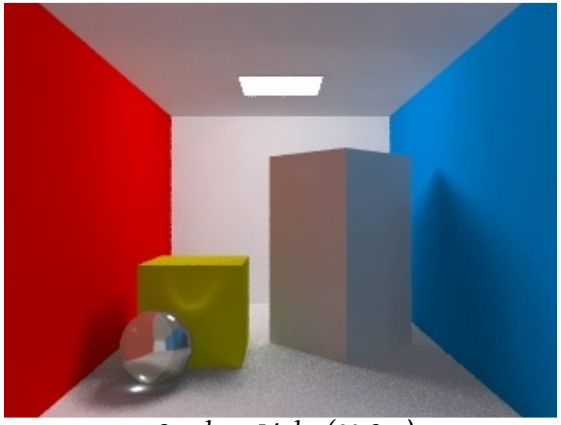
Final (13 Min 37 Sec)



Outdoor LQ (32Sec)



Outdoor HQ (12 Min 11 Sec)



Outdoor Light (20 Sec)

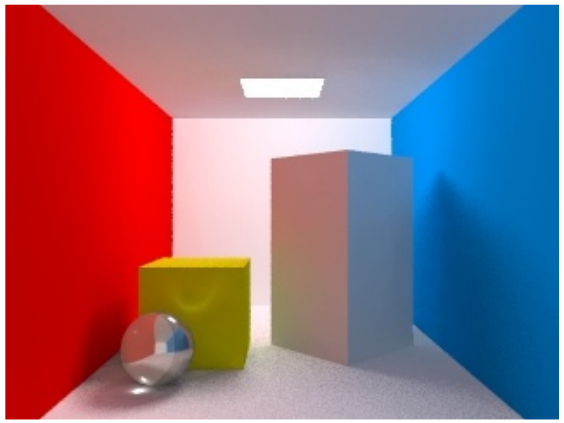
Above comparison of the quality and render times with different radiosity on the same scene:

You can achieve better radiosity configurations by modifying the radiosity parameters inside Povray or, by modifying the PovGlobalRad value from within Blender. Remember, whenever you modify Radiosity parameters in Blend2POV you will have to generate new Bake Radio&Phot.

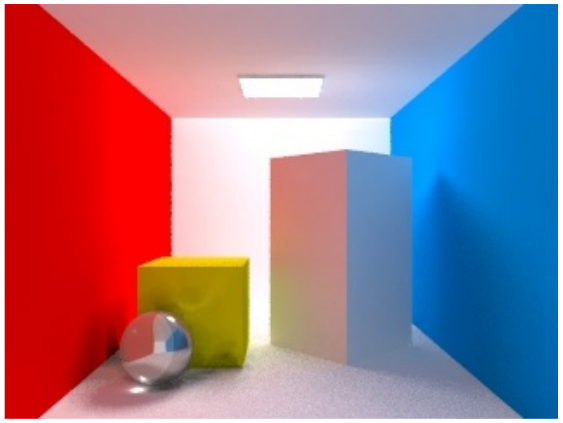
Photons

This button enables the generation of caustics in your scene. If not activated by default, fake caustics are generated, whose results can sometimes be good. The parameters of the photons for caustics are stored in the "exp.ph" file. The photons are used as helpers for Radiosity calculations.

Although the fake caustic results are not so great, these can be useful in scenes with large quantities of water (for example, underwater in a sea or a pool). In these type of environments, rendering the caustics with photons will take a very, very long time. So, it is suggested to use fake caustics here. This can be directly adjusted from Povray using the keyword caustics power, this command can have a value between 0.0 and 1.0. Note that caustics are not simulated when the value is set to 0.0. The value of caustics power is introduced as an Interior parameter of the materials.

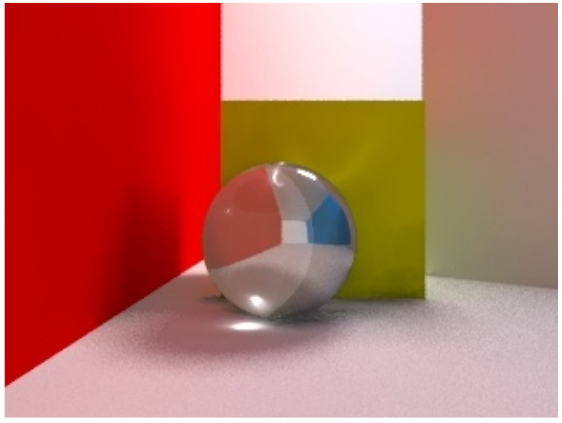


Indoor LQ (46 Sec)



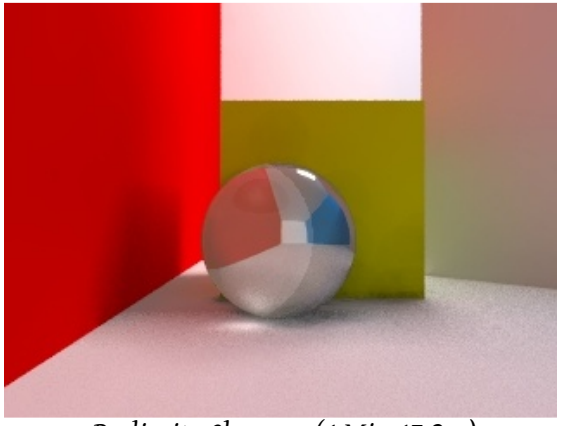
Indoor HQ (25 Min 36 Sec)

With Photons



Radiosity 2bounce (11 Min 34 Sec)

Without Photons



Radiosity 2bounce (4 Min 47 Sec)

Jitter

This is a condition for the area light that makes it generate a random sampling of the area light, thereby smoothing the layers of light & shadow samples. This, however, causes problems in animations as it generates random samples of the smoothness and seems to jump in the animations. See the example images for a better understanding of its effect.

Default Ambient

This produces the same effect as that of Blender's 'Emit' function.

Exec. PovRay

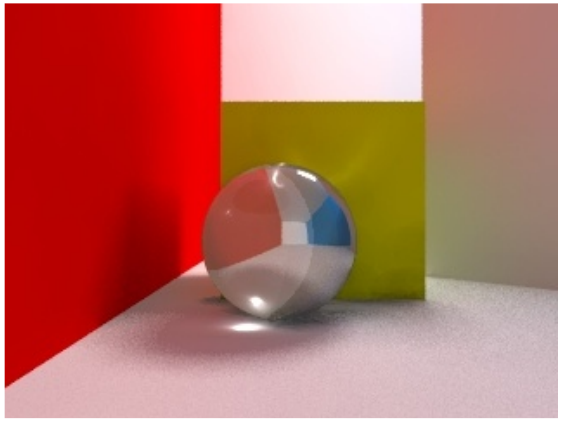
The function of this button is very simple, to run PovRay for rendering as soon as render button is pressed. If Exec. PovRay is activated, the scene gets exported to the Povray format and the image is rendered directly within Povray.

Auto AA

If this button is pressed, the render is Anti-aliased in Povray. It is ideal for a test render. It is pressed by default. If you deselect this button, three new options will appear: AA Samples, AA Passes and Thr (AA Threshold). These new options will allow you to have manual control over anti-aliasing.

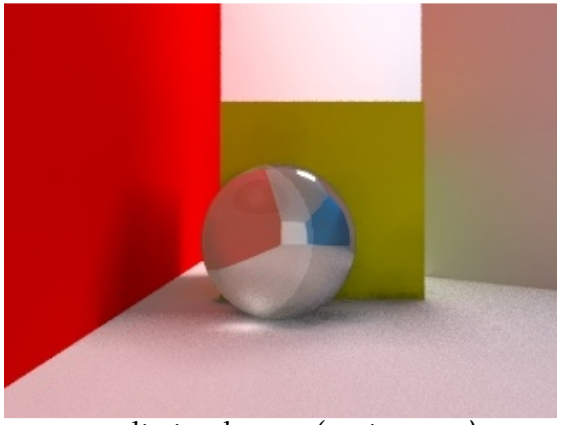
The AA Samples is used when using DOF (Depth of Field). AA Passes is the quantity of passes that are used in the render for aliasing. Thr (AA Threshold) is the separation value of the pixel region to anti-alias.

With Jitter



Radiosity 2bounce (5 Min 14 Sec)

Without Jitter



Radiosity 2bounce (7 Min 55 Sec)

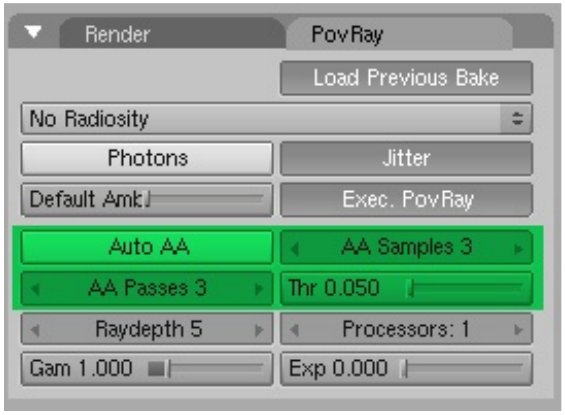


Fig5. Antialiasing options

Raydepth
This is the number of bounces of the ray and is important for glass and mirrors. Higher numbers equal better results. This also will effect the rendering time.

Processors
This option is currently not active in this version of Blend2Pov.

Gama
Gamma is related to the OSA procedure, the oversampling of pixels that are mixed to generate the final render. The conventional rendering has a Gamma of 1.00.

Exposure
This addition simulates a decrease of reactive chemicals in film emulsion during exposure. The areas that have received more light react less than other areas. Useful for high contrast scenes as it brings out the dark areas and limits bright areas.

Conclusion
I hope this basic explanation of the Blender2Pov parameters will get you started with using the awesome PovRay raytracer from within Blender. And, of course, as Blend2Pov develops, you will be able to use even more powerful features of PovRay with minimum fuss from within Blender ■

blende2povray download.
http://www.graphicall.org/builds/builds/showbuild.php?action=show&id=116&PHPSESSID=b4c1a7c03795db2af8efe72e03496217
Coder: RCRuiz

Important

To use Exposure you will need to have Megapov installed, since Povray doesn't support this function. If you use it while rendering, Megapov will be executed automatically.

Character Design 2D Sketch to 3D

by - Rogério Perdiz

Introduction

This article focuses on a very important part of 3D Character Modeling known as the 2D Sketch (or Concept Design). The sketch artwork is the basis for the successful 3D character that everyone hopes to have in their animation. It's during the Concept Design phase that we define every aspect of the character; providing a well-structured form and idea which, if properly followed, is difficult to lose while 3D modeling. Otherwise, we might turn our character into some kind of deformed alien, which may only be helpful if we do intend to make one. ;) Above, you can see one good-looking character sketch but, it is not very helpful while 3D modeling. So, for that reason I have decided to do a little sketch tutorial about what goes into making a successful 3D character starting from the 2D Sketch phase.

The Character

The stylized fictitious guy in the picture is Orion. He volunteered himself to illustrate what will be described in upcoming paragraphs. For this, you will need a digital tablet or the traditional paper and pencil.

Concept Design vs 2D Sketch

Sometimes, the gap between these two terms is blurred, thus easily misunderstood. Concept Design refers to the actual creative process of thought or idea and the design process that suits the final outcome. It is usually represented visually through continuously improved sketches or drawings of some object or character from the very first idea until the completion of a production-ready drawing. It also includes the scene details like environment style, character specialties, lighting and mood of the scene and the texture style as well.

The 2D Sketch is a single part of the Concept Design process. It gives you the basic form and hape to help materialize the idea that will gain, over time, a more organic shape on the paper. It can also be called a rough pre-visualization of the final idealized drawing.

Time-Saving Tips

Unless you're a very gifted artist with the ability to draw quality sketches lightning fast, you will need lots of time and a flexible deadline. Remember, since this is a sketch for pre-visualization of a 3D model, you may choose to be less descriptive on the paper.

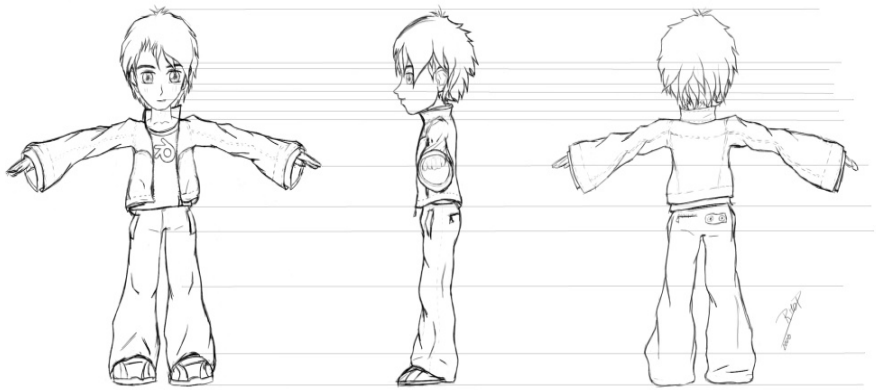
If you are drawing the sketch digitally, use Layers in your favorite paint program. Use each Layer for separate things. For instance, put the head in Layer 1, the hair in 2, body in 3, clothes in 4 and so on.

Using Layers can sometimes be tricky but, in the end it will benefit you quite substantially, saving you precious time.

The Sheet

The first thing you need to remember is that when you finish the 3D model, you'll also need to Rig(*) it so the character can be posed in the scene. It is highly recommended that you pose your character with the arms and legs slightly spread apart. You want to do this because, for instance, if the right leg is very close to the left it will be a mad job assigning the vertices to the bones. This type of pose facilitates the armature bone positioning and makes it a lot easier to model and work with the character's pose.

For the main character sketch, your goal should be a 3-sided character sheet (Front, Side and Back). This is a very crucial step for the 3D modeling phase of your character. You need to define the 3-dimensional features of the character in the sketch like muscular volume and the striking contours. After that, if it's possible, you can define more detail on a clay model, but for now your 3-sided character sheet should give you an acceptable 2D flattened reference. This should be enough to get you going. In addition, if you wish, you may also do some perspective sketches of your character with relaxed poses. I use the perspective drawings for additional reference. I always use a 3D block model made in Blender as my reference for the perspective poses. It gives a natural camera view feel to the perspective drawings and it´s really easy to define the volume of the character.



Sketching Up

When I used to see drawings made by professionals, I thought to myself "Whooo!! I'll never be able to make that, these guys are gods and simple mortals like me will never reach them." Well, what happens is that we usually only see the final drawing, and ignore the steps that they've used to get there. You can compare drawing to the construction of a house or a building. In the end, it looks pretty. But while the artists are working, it's just a lot of dirt, trash and individual parts.

Rigging

Placing an armature or bone inside the mesh to enable the manipulation of the character's pose.

The Golden Rule of 2D Character Sketching

Never forget that there is a head under the hair and a body under the clothes. If something is wrong, you will instinctively know it. You may not know exactly what's wrong but, you will know that something isn't quite right. You'll be fine if you keep that in mind.

I'm not going to describe step-by-step how to draw, but if you find my sketches great then you really are in trouble and I recommend the following sites that have quick tutorials on 2D character design. They are focused mainly on the Japanese anime style but, the techniques described are flexible and easily adaptable to any style (dedicated drawing books are preferable but, if they are out of your reach, these sites are good ones and cover well the basics):

http://www.howtodrawmanga.com
http://www.bakaneko.com
http://www.polykarbon.com

The Front View

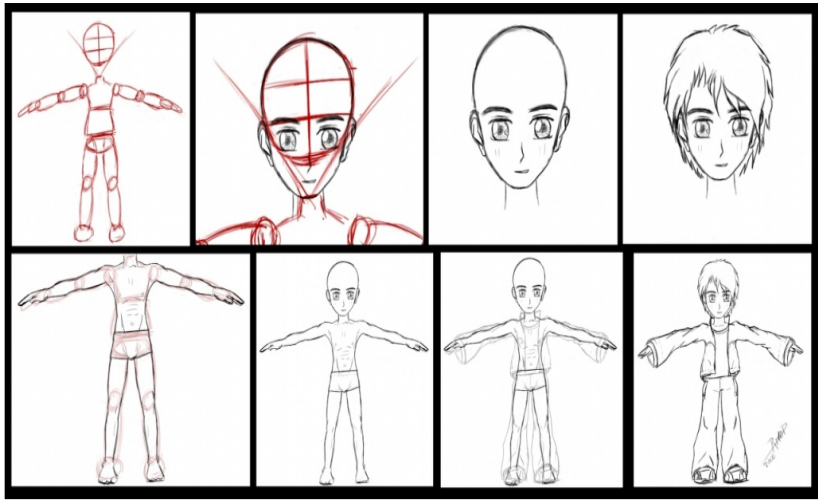
Now this is where all the fun begins! Ladies and Gentlemen! Start the GIMP (or whatever your favorite 2D image editor happens to be) or just grab a bunch of paper and your favorite pencil. The most important thing, if you're using an image editor, is to USE LAYERS, SEPARATE THINGS, and DON'T DELETE ANYTHING.

If you're using pencil and paper, also maintain the things separated. Use a light table or cell paper to draw, for example, the hair on top of the head. When you're done, scan everything and join them in the computer.

Like in 3D, the first thing you need is a block model, in this case a block ovalized drawing. This way you can define the area and position of your character on the paper and avoid boring things like, after almost completing the character, not having enough room left on the paper to draw the feet.

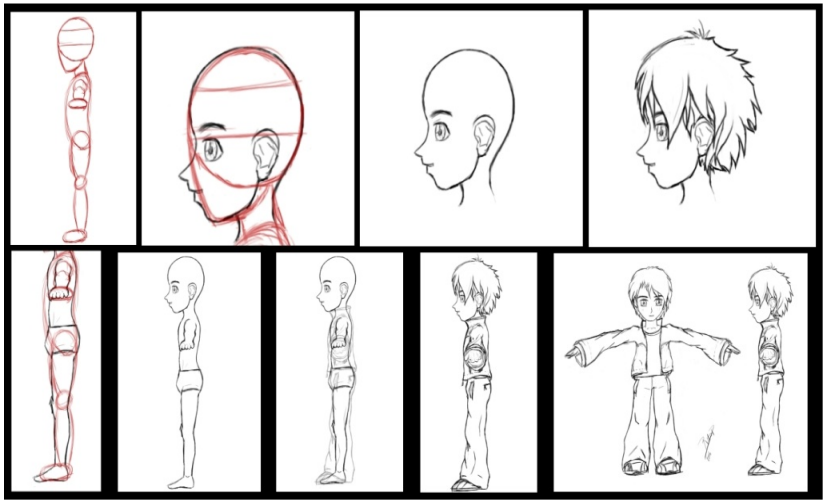
Using simple circles and ovals is generally a good method to define the body proportions, but is far from being the only method. Use circles to represent the head and the joints like shoulders, elbows and knees. Use ovals to connect them.

Next, draw the head hairless, add the hair, draw the body and finally cover him with some Cothes. Never forget to put each thing on a different layer.



The Side View

When you finish the Front view, it's time to make the Side view

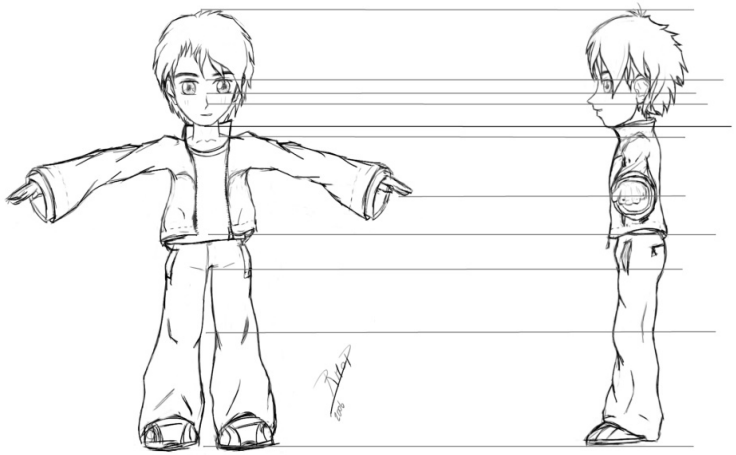


(Image left bottom).
Now here, sometimes things can become tricky. What happens is that the Front view must coincide, as perfectly as possible, with the Side view. That's called Foreshortening.

Foreshortening

Foreshortening isn't easy to say the least. Whatever you're using, the GIMP or paper, you'll need to draw horizontal lines from key parts of the Front view and use them to draw your character from the Side view. In the GIMP, you can just activate the View Grid function but, it's a lot harder, time-consuming and error-propitious.

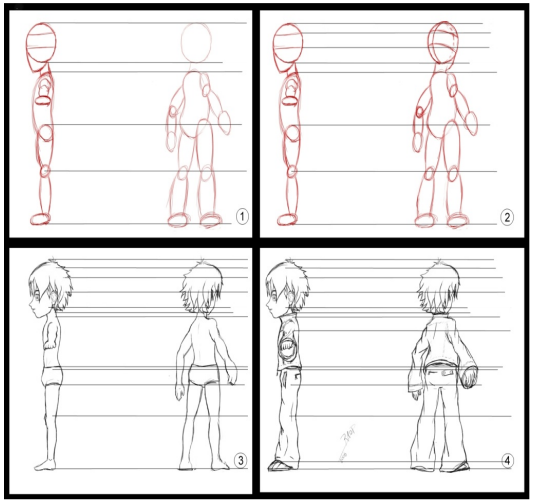
Press the [shift] key while choosing two points in the GIMP or use a ruler on paper.



Oh My! So Many Lines!!!

Don't panic just yet. Its just like a house, everything is made progressively. I've made the image below to show you the steps I've taken and that it's even useful with a different pose.

First, you only trace lines from the key parts of the block model like the head,



waist, knees, feet (1) then, you trace some more to refine the head (2) and so on...draw hair, draw the body (3) and dress him (4) just like the Front view. Don't forget to use Layers.

Perfection here is the key. The more perfectly you can do it, the easier it will be to model the character in 3D. I think that's why I always have so much trouble in the modeling phase.

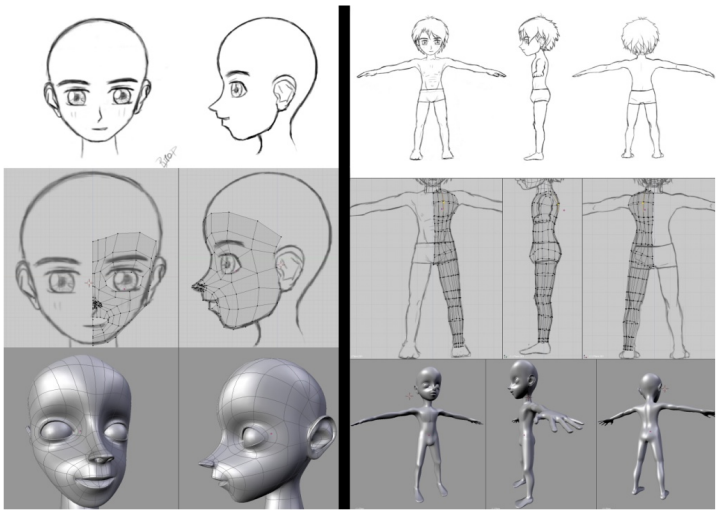
The Back View

To create the Back view you use a fusion of the methods you've already used in the Front and Side views. There's nothing more to it than that. After finishing the Back view, you're ready to fire up the Blender thrusters!

Ready to Blend

When you finish drawing, it's time to save the pictures to JPEG files. This saves space and is Blender-friendly. Any other file type you prefer is OK as well, as long as Blender can recognise it (don't go with BMP's).

Here is where the Layer division process pays off. The first thing you'll want to do is model the head (I always prefer to start with the hardest thing because then, it's all downhill from there. But, that is just me, feel free to start wherever you like.). Start by just selecting your hairless head's layer and make everything else invisible. Save the image to something like headGuide.jpg and, using Blender, open it in the Front and Side viewports (in the 3D viewport header, press *View|Background Image...* and then click the Use Background Image button) . You can adjust the position to focus only on the one that matters. Setup the Front viewport to use the Front view sketch and the Side viewport to use the Side view sketch as background images.



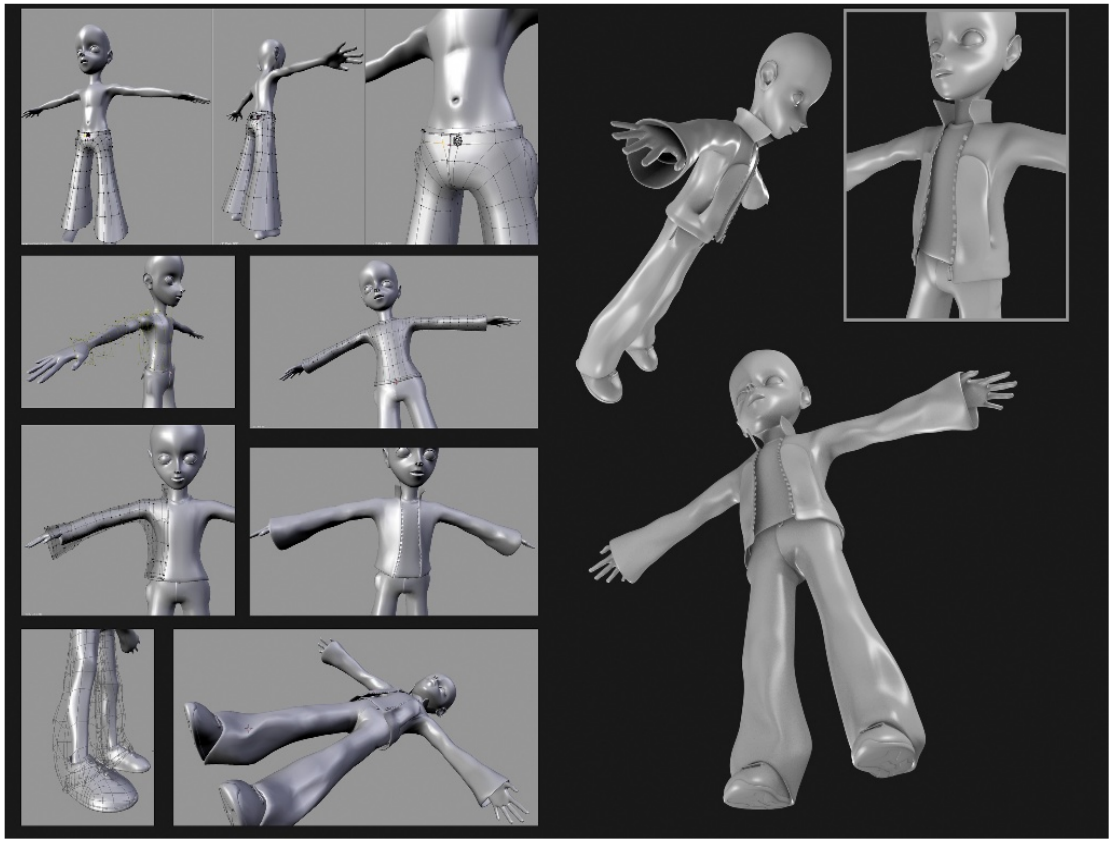
After you model the head, you'll want to do the body. So, select the Front, Side and Back body layer and make sure that everything else is invisible. Save the image to something like bodyGuide.jpg and blend on. For the clothes, you also can and should use the same process.

Closure Notes

Remember, it's always preferable to do a sketch before going to 3D, even if you're drawing skills are not very developed. You'll find a lot of drawbacks, anatomical issues and annoying complications in your character design during the 2D sketch phase that you, for sure, don't want to find while modeling in 3D.

Note that I´ve only shown you how to draw a full character. Sometimes it's enough but, if you really want to specify details, it's better, or is even required, to do sketches of only the head, the feet, some specific article of clothing, etc. that occupies an entire page. However, the method is pretty much the same.

Well, it's the end of the journey with me. I hope this tutorial was somehow useful to you and prepares you well for all the cool ways to do 3D character modeling that are in this edition of blenderart magazine... I'm in fleas to read them. Ho! About Orion... well, perhaps you'll see him again someday. Until next time... ■



3D Views of Orion

MakeHuman
"Making" Humanoids Since 2000!

Manuel Bastioni (English version)
Alessandro Proglio
Antonio Di Cecca
Giovanni Lanza
Martin Mackinlay

What is MH?

MakeHuman is a software package, written completely in C++ and available for all the main platforms (Windows, OSX, Linux, etc..). Its aim is the modeling of three dimensional humanoids.

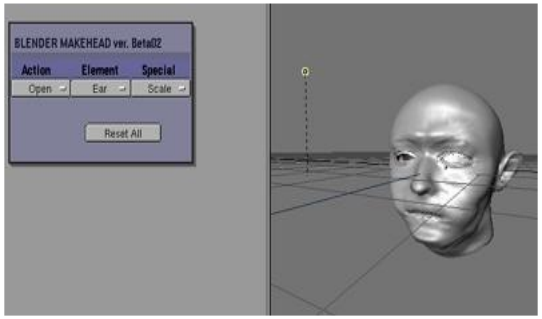
Users can set up the age, weight, sex, race, size of the nose, shape of the face, proportions of arms and legs, and a whole series of other well defined parameters, so we can say that MH is an artistic/para metrical modeling tool.

Every feature is defined by choosing it's percentage, and all the features are "added" to each other to obtain endless combinations of forms. This allows the creation, with a few clicks, of extremely realistic characters, ready for use in a variety of professional graphics applications.

Short History

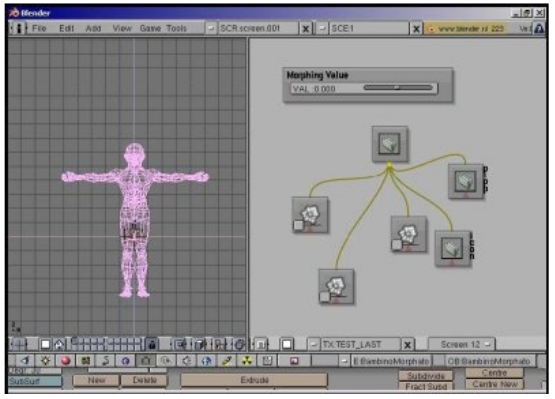
The ancestor

MakeHuman was born, near the end of the year 2000 in an Italian community of open source graphic software developers. Manuel Bastioni had previously developed something similar: a python script for Blender, which modeled only the characters' head. It was a simple tool called "MakeHead", and based on Blender's vertex keys, which can be considered, in every respect, MakeHuman's ancestor.



First script for Blender

Manuel's new proposal was to do something more flexible, which allowed modeling an entire character, from head to foot. Others liked the idea, and after about four months of development, thanks to the first group of programmers, in particular, Filippo di Natale and Mario Latronico, the first alpha version was finished and immediately released under the GPL license.



Others were responsible for the translations (Fabrizio Cali', Lorenzo Daveri), for logos and other aspects relating to the material's publication.

First model

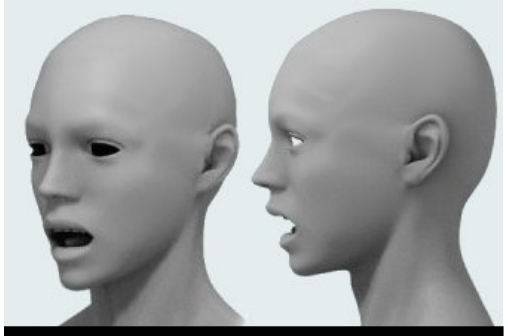
Even if the GUI was defined, and the morphing code was mostly ready, there was another delicate and important problem: the realization of base humanoid, which had all the necessary characteristics for it to be transformed into any character, and which was also an optimized mesh, from which could be created the character of a movie or a low resolution character suitable for video games.

Furthermore the basic mesh had to be sexless, ageless and race-less, just to make things a little more complex. After much discussion, it was finally modeled by Enrico Valenza. This was the first version of the model, which was be used up until two years ago, as we shall soon see. After some time, the project was officially recognized by the Blender Foundation, by supplying important new tools, like a mailing list, a specific forum, bug tracker, and was included in Blender's project page.

Unfortunately, due to internal disagreements, the collaboration was reduced considerably, and the official site was moved to dedalo-3d. After a short period time, MakeHuman had an important modification. The passage from an old complex GUI to a lighter one was finally completed, based on the input from a console written in python by the project administrator. This made way for the release of version 1.6.

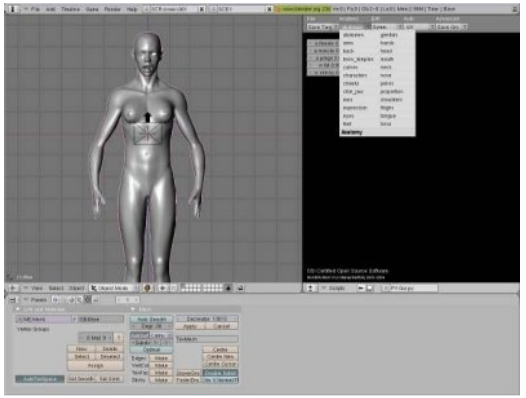
After this release, the participation of the earlier developers, with the exception of the administrator and a few others (Lorenzo Daveri and Cicca), was reduced drastically, and the project seriously slowed down. Fortunately, it was only a period of transition in which MakeHuman was going to adopt a more international look thanks to the contribution of Craig Smith on modeling, to the contributions of Olivier Saraja, to the code of Michael Schardt (import-export verts group) and to the system of bones from Andrew Kator.

Thanks to these new aids and numerous new features, a new version was released which jumped directly from version 1.6 to 1.8 in a short period of time.



Version 1.6 and the Blender Foundation

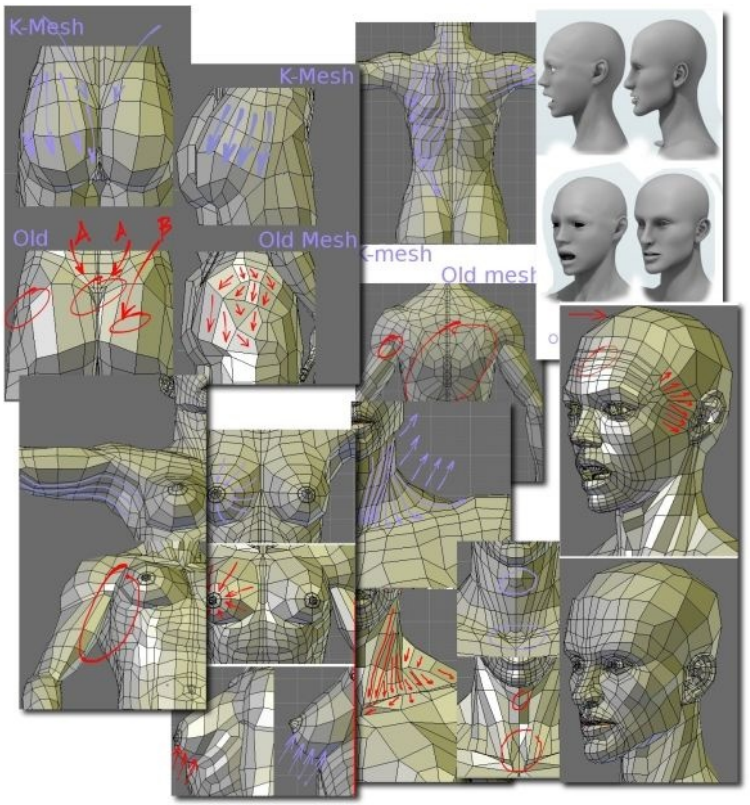
The stable version in Python 1.6



FACS
Due to the success of this new version as well as to the mediation of Craig and Lorenzo, the project received an important donation which consisted of an entire documentation of 20 years of research of the dott known as FACS. This so-called FACS (Facial Actions Coding System), allowed for study of an entire system of coding of facial expressions. The donation came from Paul Ekman, who was a psychology teacher in the department of psychiatry at the University of San Francisco.

Version 2.0 in python, unstable
At this point with version 1.8, the development of the python script substantially stopped. Two other versions had come out and was written almost entirely by a new developer, named Paolo Colombo, that introduced an innovative GUI, better than the previous one. But that unfortunately remained in the experimental phase due to some compatibility problems with Blender, OpenGL, and video cards. These drawbacks caused severe delays that Paolo had to rewrite the GUI several times in order to adapt it to the changes of the python API of Blender and to avoid problems with the video cards.

The new mesh
Version 2.0 of the script was conceived in a new manner that was optimized for professional use and never succeeded in moving beyond the unstable phase. Even though it wasn't fully usable, it did have a very important new feature that retained after months of work: a new basic mesh. The first version had in fact begun to show it's limitations as meshes that were too optimized (about 7000 vertex), did not succeed in covering all the required morphings, and was lacking in some fundamental edge loops for the facial and muscular animation. For this reason, a new modeler, Kaushik Pal, already author of exceptional humanoid models for Blender and Maya (CGtalk thread), created a new mesh. While not using an excessive number of verts (about 11000), the model kept up with all the professional requirements which was the result of studies of more well-known meshes. In the following images the main differences between the two meshes are illustrated (the "critical" points of the old mesh are marked in red).



Despite the fact that Kaushik's mesh were already high level, the MakeHuman team wanted to present the model also to the attention of famous modelers. Among these artists, of world-wide reputation Steven Stahlberg and expert subdivide modeling TamÃ s Varga just to name a few.

Here we quote the words of Mr. Stahlberg, of which we are particularly proud, answering to our post (by Tom Musgrove):

"Tom, that's a great resource for artists who want to study topology. No real crits, except maybe you could change the direction of edges down the cheek, to easier incorporate the infraorbital fold (which is one of those things that everyone has, although it's very subtle with some). To comment the face, I think the red and blue loops are really important, but the brown and green ones don't really need to be perfect loops. They can be, but there's no pressing need imo"

Original thread

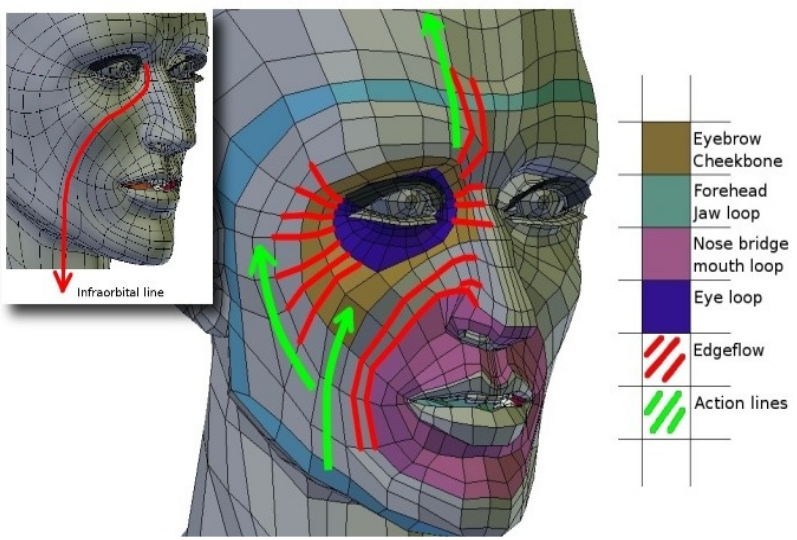
Few words, but very gratifying for the team. Obviously then the infraorbital line has since been put into place, as well as the other suggestions that came out during the discussion.

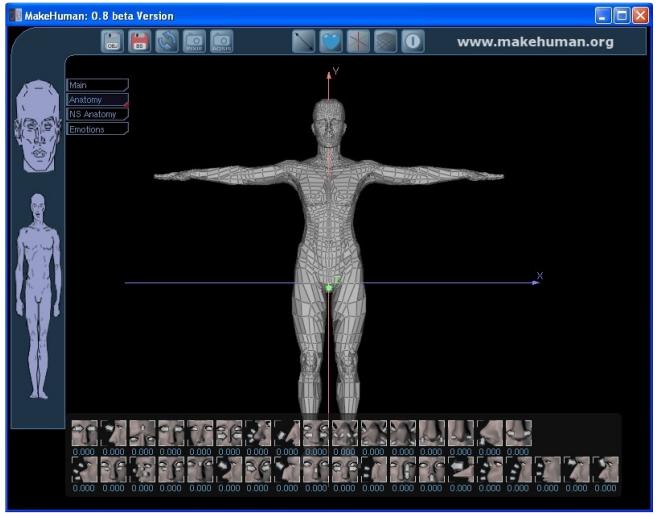
At the end of the work, we have collected some explanatory images, particularly concerning the facial topology, and in which the so-called lines of action have been highlighted.

The turning point: the standalone version in C

However, even with the new mesh, version 2.0 of the script was still limited by the problems described above. Finally we realized that the functions, performance, and stability we wanted to achieve weren't possible with MakeHuman as a python script. It was necessary to "move" it outside Blender, and to rewrite it from scratch in high-performance languages, such as C or C++. The first step of this major transformation was possible thanks to a major contribution from Paolo, who alone, and in very little time had written a new MakeHuman completely in C, multi-platform, and with an innovative interface.

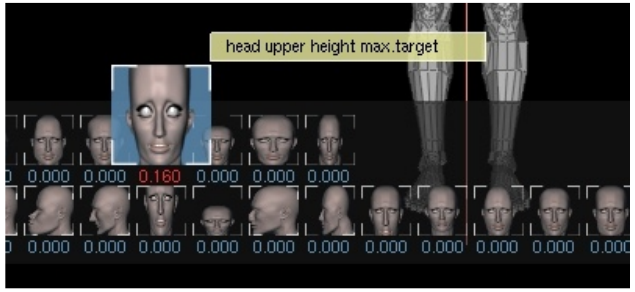
We'd like to mention in particular the "slider-images", a new button which auto zooms on mouse-overs (working as a preview of the target) which permits holding the mouse button to set the percentage of morphing. Since this was a completely different version from the python script and we had rewritten it from scratch, the standalone version had a lower version number: 0.8a.





Downloading of the standalone version of MakeHuman can be found using hosting provided by SourceForge.

In this way, the python version hosted by the Blender Foundation, remains separate from the version written in C. The version currently available on SourceForge is still Paul's version, but with some additions and bug fixes due carried out by other programmers (OBJ exporter by Andreas Voltz, OSX porting by Tan Meng Yue). End users found the program to be quite usable, but programmers continually worked to further advance the code, in hopes of significantly improving the usability for the end user. We then arrived at a new version that's going to be released.



Next version: MakeHuman 1.0 (beta), in C++

Animorphs

While we were working on MakeHuman version 0.8, Andreas Voltz, a student of Applied Computer Sciences at the University of Fulda, in Germany, was developing some concepts of the python version in order to construct a new, more flexible, library that was platform independent and written entirely in C++. His aim was to use it for the software that he was preparing for his bachelor thesis called "MeasureHuman". He wanted to not only modify aspects of 3D humanoid based on targets like MakeHuman, but also on user input measurements.

Andreas had gotten in touch with the team, announcing that after receiving his degree, he would make his libraries available under the GPL. This was great news; the libraries were well written, highly object oriented, and written to be easily extensible and modular. The C version on the other hand, was developed in a procedural style, and wasn't so easy to update and change. The passage from C to C++ meant offering the users a still better product, with a more standard and comprehensible code, written very professionally and incorporating all possible cautions to satisfy the rules of OOP programming. Certainly, this would have caused some delays because, once again, we should have rewritten the code from scratch. Though, it was worth the trouble since subsequent development of the application would have been more slender, professional and reliable.

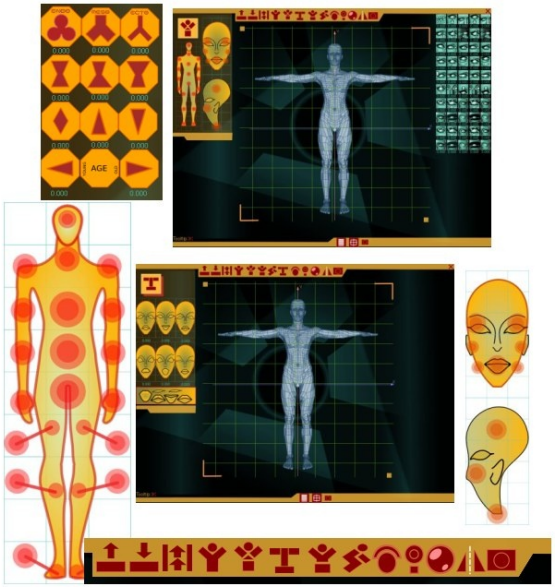
MHGUI

Once it was decided to use Andreas' library, which he dubbed "Animorph", we needed to write a simple and stable GUI.

In line with the philosophy of the project, we avoided the use of large existing libraries. We wanted MakeHuman to be compact and an extremely reliable tool with few dependencies, and with well defined features. After having reviewed a large number of existing libraries, we finally decided that, for the special interface of MakeHuman, the best thing would be to write the necessary classes ourselves. Simple = Reliable, Simple = Easy to manage and update. For these reasons, the GUI contains only what is necessary to us and we are doing everything to test it thoroughly. The core has been written with the help of Ninibe Labs. A particular thanks goes to Simone Re and Manuel who together have developed the first sketch of the GUI. Subsequently, everything has been reviewed by another programmer, Hans Peter Dusel, who has also worked on the porting for OSX.

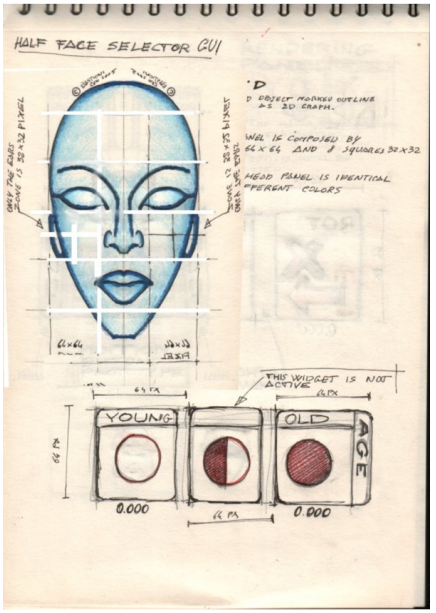
MakeHuman

Now that finally both Animorph and MHGUI have reached a satisfactory level (even if is our intention to continue to perfect them), the only thing that remains is to combine them into a single application, MakeHuman 1.0. This will be a beta version until it is fully stable and has the necessary features to to be considered "complete". The following images are a preview of the work that we are carrying forward.



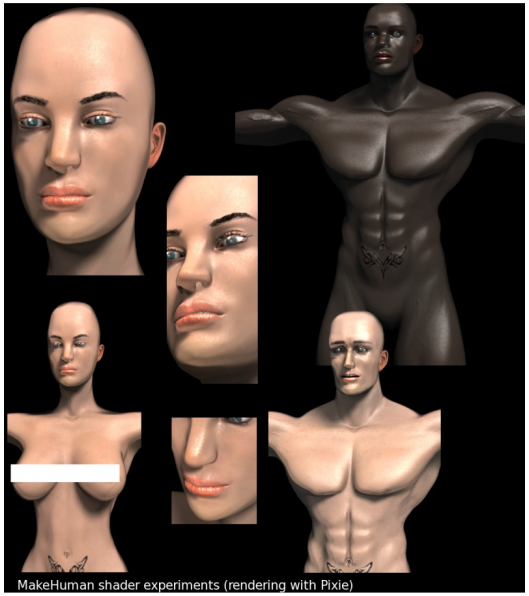
The new interface, although it uses the ideas from version 0.8, has been carefully planned to have a simple and intuitive design. Each element has been carefully studied to be easy to understand and use.

Even if the GUI will continue to be improved to become even more intuitive, we are still going to be preparing adequate documentation. This is possible thanks to a team exclusively assigned to the production of documents, their translation and logical organization. A special thanks goes to Alessandro Proglio for translating the documentation and administrating the Wiki (still in phase of development), to Martin Mackinlay for the final revision of the ITA-ENG translation, and to Antonio Di Cecca and Giovanni Lanza, who actively contribute to the ITA-ENG translation.



Output

MakeHuman is intended to be a very specific tool, limited to a precise field: to rapidly model professional 3D characters with all necessary features for animation with the use of suitable software (Maya, Max, Blender, etc...). Therefore it is a tool that finds itself ideally positioned in the professional pipelines, and that does not want or have to be an end in itself.



MakeHuman shader experiments (rendering with Pixie)

Additionally, to a limited extent, we want MakeHuman to reach an equally professional level with static poses and to be able to provide photorealistic renderings of the subject. On one hand, it can serve as a preview of an eventual pre-production (of games, movies, etc. ..), yet on the other, can be used by the artist to produce final images (posters, gallery, etc. ..). Therefore, in conclusion, the output of MakeHuman can be divided in two large categories:

1) Exporting in the most common formats, compatible with the widest range of software.
2) Photorealistic (but static) rendering of the characters in pose, complete with hair and clothing, suitable for previewing and the production of images.

Currently only export in OBJ format is supported, yet even though OBJ has become a standard, we have intentions to add additional formats. Many features have already prototyped in python, for example â€œhairâ€, which only needs to be translated into C++ and adequately incorporated in the GUI. With regards to rendering, we are working on a skin shader engine thats Renderman compliant. Here are some images made with Pixie.

Help from the community

Modelers

Any capable modeler that knows the basic features of Blender are invited to join and help us. Building a new morph is very simple and does not require any programming knowledge. It is only if you want to export a ". target" file. For example, try it directly in MakeHuman. A special script for Blender, called "MakeTarget", is required and may be downloaded from the Blender Foundation's project site .

It's a very interesting tool because it allows you to also manipulate the ". bs" formats. A short introduction to its utilization can be downloaded here.

Programmers

Help on the developer's part is also welcome even though, obviously, the participation is bound to specific rules and allocations of the tasks by the administrator. At the moment, we need an expert Windows programmer as most of the developers work with Linux, and therefore, the Windows version requires a maintainer to monitor optimizations, porting of the libraries, and checking of the installer, etc ■

Split/Frame rendering on ResPower Super/Farm.

by - Cory King and Early Ehlinger

Outline:

I - What is Split/Frame rendering?
II - How does ResPower accomplish Split/Frame rendering with Blender?
III - What about re-assembly?
IV - How much speedup does Split/Frame provide?
V - Real world examples.
VI - Known Issues.

I - What is Split/Frame rendering?

ResPower is constantly searching for ways to speed up the process of 3D content creation, specifically in the area of rendering. Speeding up the render times of animations with hundreds or thousands of frames is fairly straightforward; simply render each frame on a different computer so that multiple frames run in parallel.

Split/Frame rendering is basically the same process, only it is designed for still frame images.

The basic idea behind Split/Frame rendering is to break a single frame into multiple chunks called "buckets". Each bucket is then rendered on a separate computer so that several (possibly one hundred or more) chunks can be run in parallel. This speedup is linear in theory, meaning that a frame split into 100 buckets should render 100 times faster than the same frame rendered as 1 bucket. Unfortunately, because of Amdahl's Law, this is not the case, although the speedup can still be quite remarkable. Check the sidebar for more information on Amdahl.

Split/Frame rendering is not a new concept and has been available for many 3D rendering engines for years. Professional 3D packages such as 3DStudio and Lightwave support Split/Frame natively, but even with one of these packages the advantages of Split/Frame are only seen when using a render farm like ResPower.

II - How does ResPower accomplish Split/Frame rendering with Blender?

The Blender Python API has facilities to perform a "Border Render." This basically means that a rectangular section of pixels from any image can be selected and the render engine will only render the pixels within that rectangle, leaving everything else black. Using this method, ResPower can split an image into any arbitrary number of buckets, and render each bucket on a separate computer. Then, using another open source software package called ImageMagick, the frame is automatically cropped to remove all the extra black space. Each individual bucket's output is saved as a separate image in the user's Renders folder.

The ability to render separate portions of a single frame in parallel allows 3D artists who work with still frames to take full advantage of a render farm. With this technique, a single frame can be rendered at full resolution in a matter of minutes at ResPower where it might take hours, or even days, to render at the same size and quality on a single computer.

On top of that, an artist can continue to tweak his scene, or work on something completely different while his frame is rendering because his machine won't be bogged down with the arduous task of rendering.

III - What about re-assembly?

ResPower has recently introduced the ability to re-assemble all the buckets from a Split/Frame render with a process we call "stitching". For frames submitted as Split/Frame, there is a command on the jobs page called Stitch. Stitching once again uses ImageMagick to programmatically pull all of the individual bucket images together and append them into one full-size image. All of the individual bucket images remain available for download if the user wishes, but an additional image called "frame_x.stitched.ext" will appear in the Renders folder once the stitching is finished, where 'x' is the frame number and 'ext' is the file extension, i.e. png, jpg, bmp, etc.

ImageMagick works from the command line so that overhead associated with starting a graphical user interface is completely eliminated. This means more resources are available to devote to the process of restitching, and that means customers get the finished product faster. Another advantage of ImageMagick running from the command line is that restitching can be almost completely automated. With a single command, the farm will fetch every image required, give them to ImageMagick, and wait for the final output.

This ability is vastly superior to the old way of re-assembling bucket images where users had to use photo editing software such as Photoshop and manually put all the images together to get the finished product.

IV - How much speedup does Split/Frame provide?

As I said before, the theoretical speedup for Split/Frame rendering is linear, so that a 50 x 50 split should render 2500 times faster than no split. Unfortunately, ResPower hasn't been able to secure any real estate in Theory (prices are *incredibly* high there), and things that work in Theory don't always work here. The speedup is considerable with splits in the range of 16 buckets all the way up to 400 buckets. Unfortunately there are many factors involved in determining the best split, and trial and error is generally the best way to find the optimal split.

Because Blender is only usable on the ResPower Super/Farm by purchasing a subscription, customers don't have to worry about picking the wrong split. If a 2 x 2 split isn't fast enough, you can try a 4 x 4. If you start with 10x10, and you've passed the point of diminishing returns, you can back it up for a 8x8 split.

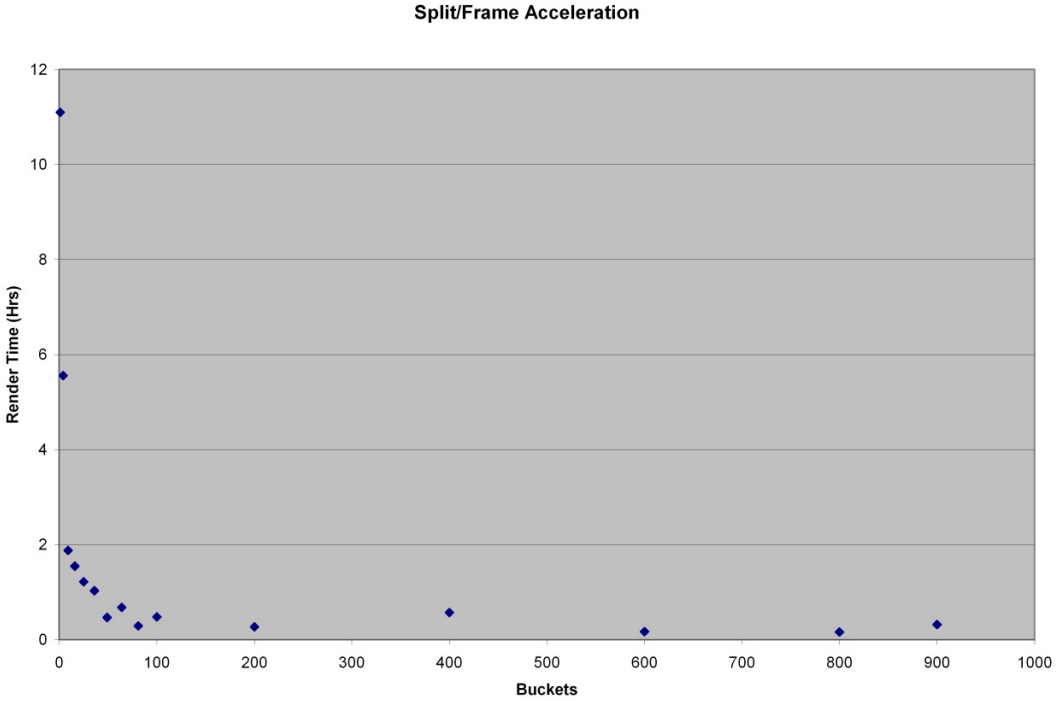
V - Real world examples.

The file used for determining these times can be found here. It is provided so you may render on your own system to determine the speedup you can expect when using Split/Frame on the ResPower Super/Farm.

http://www.respower.com/~CoryKing/split_frame_test/

These are the times required to render this file at different splits using only the fastest rendering nodes available on the farm. The resolution of each rendered frame is somewhere close to 2048 pixels x 2048 pixels. The slight changes are due to the fact that the width and height in pixels must be a multiple of the number of buckets for stitching to work properly. For instance, an 800 pixel x 600 pixel frame should not be rendered with a 7 x 7 split because 7 doesn't divide evenly into 800 or 600.

Please note that render times depend on several factors including scene complexity, network stress, the availability of render nodes, and more. Because of these factors, this very small set of data is fairly inconsistent. A more robust data set would show a smoother curve of progression between data points ■



1 x 1 (no split) - 2048 pixels x 2048 pixels
render 11 hrs 06 min 46 sec

2 x 2 - 2048 pixels x 2048 pixels
render 05 hrs 34 min 08 sec

3 x 3 - 2048 pixels x 2048 pixels
render 01 hrs 53 min 11 sec

4 x 4 - 2048 pixels x 2048 pixels
render 01 hrs 32 min 55 sec

5 x 5 - 2050 pixels x 2050 pixels
render 01 hrs 13 min 21 sec

6 x 6 - 2046 pixels x 2046 pixels
render 01 hrs 01 min 11 sec

7 x 7 - 2051 pixels x 2051 pixels
render 00 hrs 27 min 17 sec

8 x 8 - 2048 pixels x 2048 pixels
render 00 hrs 40 min 53 sec

9 x 9 - 2052 pixels x 2052 pixels
render 00 hrs 17 min 17 sec

10 x 10 - 2050 pixels x 2050 pixels
render 00 hrs 27 min 54 sec

10 x 20 - 2040 pixels x 2050 pixels
render 00 hrs 13 min 27 sec

20 x 20 - 2040 pixels x 2040 pixels
render 00 hrs 33 min 21 sec

20 x 30 - 2040 pixels x 2040 pixels
render 00 hrs 10 min 10 sec

20 x 40 - 2040 pixels x 2040 pixels
render 00 hrs 9 min 27 sec

30 x 30 - 2040 pixels x 2040 pixels
render 00 hrs 16 min 31 sec

Amdahl vs Gustafson: a Parallel Showdown

Amdahl's Law states that a parallel process cannot achieve linear scaling because there is a portion of every program that must occur in serial. This portion determines a limit to the speedup one can hope to achieve by adding processors. If 80% of your program's time is spent in this serial portion, then adding an infinite number of processors can eliminate, at most, 20% of the program's total execution time, and that's assuming that the communications involved come at no cost.

With rendering, the percentages are typically inverted: 1% or less of a render's time is spent in the serial portion, so adding processors can give you a nearly linear speedup: going from 1 to 2 processors will give you nearly double the throughput, and going from 1 to 700 processors will give you nearly 700x the throughput. In other words, if you want to render 700 frames of animation, and have 700 computers to do it with, you can expect it to take roughly the amount of time it would take you to render the first frame on 1 computer. At ResPower, we routinely see this sort of speedup for animations.

But what about Split/Frame renders? Can you divide a single frame 700 ways and have it render 700 times faster? Unfortunately, for individual frames, the serial percentage involved is significantly higher than for animations, and you start to see a point of diminishing returns much sooner. As the charts show, you see a tremendous boost very quickly, but after about 100-200 computers, adding buckets doesn't speed things up very much.

Why is that? Well, John Gustafson created a rebuttal to Amdahl that explains it fairly well. With a single Split/Frame render, the size of the problem remains unchanged, and so your speedup follows Amdahl's Law. The shape of the curve looks almost exactly like a graphing of Amdahl's equation:

speedup=1/(s+p/N)

With animations, we tend to increase the size of the problem more than the number of processors. As a result, you see a scaling mode where 400 frames finish in roughly the same amount of time as 1 frame. You haven't increased the speed of any individual frame - you've just done more frames. So, was Amdahl wrong, or Gustafson? ResPower's testing indicates that they're both right, depending on your perspective. If you keep the problem size constant and add processors, you see a very logarithmically-shaped curve, with maximum benefit around 100-200 computers (at least for the test scene we used). If you change the problem size in conjunction with the number of processors, you see a nearly linear curve.

As it turns out, Dr. Yuan Shi over at Temple University was able to prove mathematically that Gustafson and Amdahl were both saying exactly the same thing, when you account for the imprecision in their respective terminologies.

Amdahl vs Gustafson: a Parallel Showdown

Their formulae all worked out to the same thing, with results that jive with what we see at ResPower: a static problem size yields diminishing returns, but additional processors let you solve larger problems in the same amount of time. So thanks to Dr. Shi, we can now see that the universe makes sense again and peace has been restored.

References:

http://en.wikipedia.org/wiki/Amdahl's_law
http://joda.cis.temple.edu/~shi/docs/amdahl/amdahl.html
http://www.scl.ameslab.gov/Publications/Gus/AmdahlsLaw/Amdahls.html

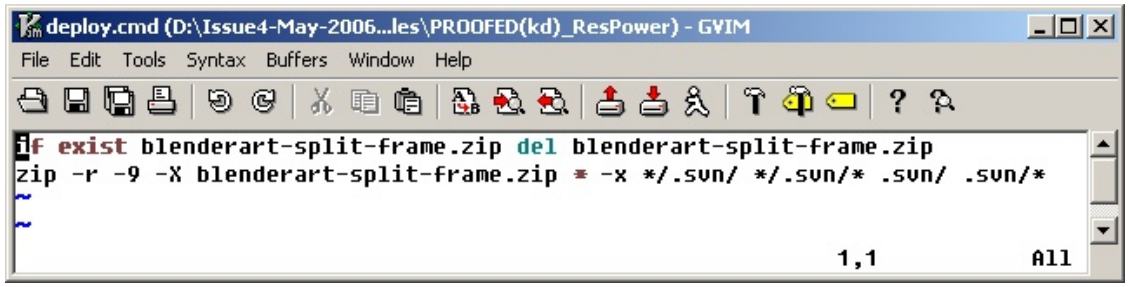
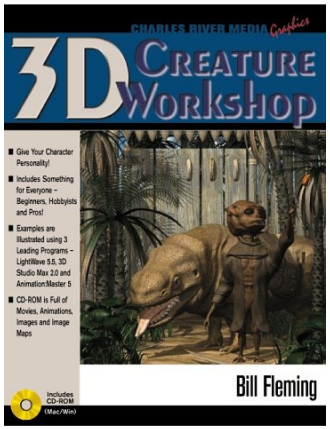


FIg2. The deploy command/script.

3D Creature workshop

Bill Fleming and Richard H. Schrand



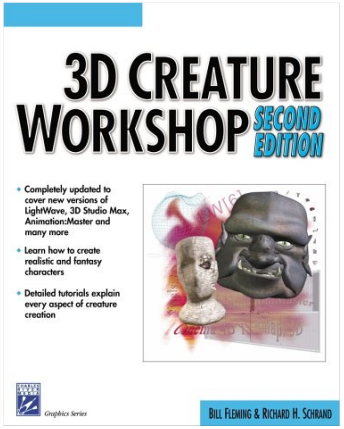
In this issue we are doing a dual book review. We will be looking at Bill Fleming's "3D Creature Workshop" and the updated 2nd edition he co-wrote with Richard H. Schrand. Both books cover the same great steps and techniques, with the 2nd edition containing four new chapters highlighting fantasy creature creation and the animation of single mesh models. A variety of new creatures are used to illustrate techniques in this edition as well.

Whether you have a copy of the first or second edition, you are holding an invaluable resource on creature/character modeling and creation. Each book starts off with a thorough discussion of creature design. Then you work through the elements of creating a creature biography that will help you with your design decisions. There is also a great section on the value of source material and how it fits into designing your creature.

The next major sections contain chapters that cover modeling in various software programs. And while that might put you off buying this book, don't let it. The tutorials are well written and while slanted toward specific programs, can be applied very easily to blender techniques and tools.

Once you have finished modeling your creature of choice, there is a rather in depth section covering texturing. Bill Fleming shows some great techniques for adding realistic detail to your texture maps.

While both editions are somewhat older, nothing new has hit the market that can surpass or replace the amount of information available in these books. If you want your creatures/characters to be realistic and appear to be as lifelike as possible, then you need to check at least one of these books out.



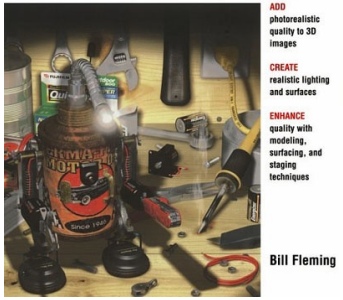
If you follow the following link, you can download a sample chapter of the 2nd edition from Charles River Media. ■

3D Creature Workshop
by Bill Fleming
· Paperback: 450 pages
· Publisher: Charles River Media; Bk&CD Rom edition (May 1998)
· Language: English
· ISBN: 1886801789

3D Creature Workshop, Second Edition (Graphics Series)
by Bill Fleming and Richard H. Schrand
· Paperback: 441 pages
· Publisher: Charles River Media; 2 edition (February 1, 2001)
· Language: English
· ISBN: 1584500212

3D Photorealism toolkit

by Bill Fleming



In this issue we are reviewing "3D Photorealism Toolkit" by Bill Fleming.

Have you ever worked long hours to model, rig, stage and light your scene, just to render it and say "something just is not quite right"? If you were not able to figure out what that "something" was, then this book may be for you.

3D Photorealism Toolkit, by Bill Fleming, provides many ideas on how to achieve that coveted "Photorealistic" look. This is not a How-to book per se. Instead, it is a group of good ideas that you can use to help improve your image creating ability – a true Toolkit.

The book is divided into five parts, each one covering a specific area of the creation process.

Part I is an introduction to the elements that make up photorealism, it is a 10,000 foot view of what will be covered in detail in the rest of the book.

Part II dives into modeling techniques that enhance the details of the models, including the use of screws, seams, bevels and fillets, etc. It also discusses the mechanics of how things are put together, as well as the materials that they are made of.

In **Part III**, Fleming discusses surfacing techniques, concentrating on fundamentals, such as aging surfaces, proper use of specularity, how to use images and procedural maps.

Part IV deals with staging techniques to include how to plan your image, what items to include, how to clutter the stage, etc.

Part V talks about using camera angles and lighting to the most advantage. Again the emphasis is on the theory of how this impacts the final image, backed with examples of how it works.

A unique item about this book is that it does not emphasis technique as much as how important it is to be aware of your environment. As artists we have to look deeply at the world to find the small details that, when left out, make our scenes look unnatural. A great example that Fleming uses to show this is including a weld between a fire hydrant and its base. Without the weld, even with all of the textures and lighting, the image still does not look right. Just the addition of a few extra polygons and the image looks like a photograph.

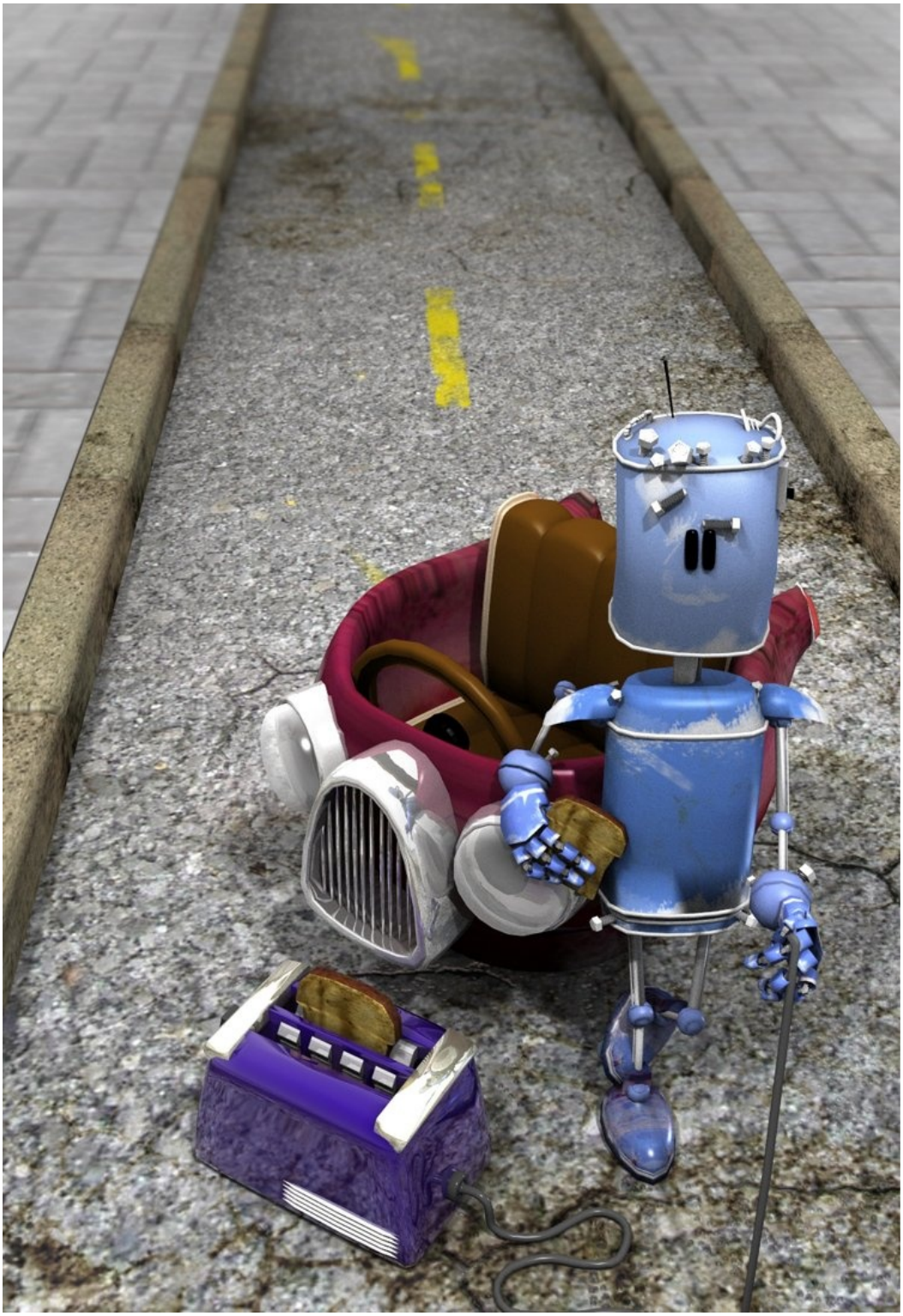
The book does have some drawbacks. It is covering a lot of ground in a relatively small number of pages, so some areas are not as detailed as one would like. The largest part of the book covers the first three parts, leaving the last two short on detail. Finally, because the book is a bit old by computer standards (1998), some of the techniques, partially for some of the lighting techniques, feel a bit dated. However, the book is non-application specific and because the techniques being taught are well founded, the reader will find that these items are not a drawback.

This is a "must read" book for beginning and intermediate CG artists and even experienced people may find some gems that they had not known or thought about before ■

Name: 3D Photorealism Toolkit
Author: Bill Fleming
Edition: 1998
Paperback: 328 pages
Publisher: John Wiley & Sons, Inc.
Language: English
ISBN 0-471-25346-4

by - Bruno Okano - "Samara"

by - Kadir Celik - "Moddy"



by - Kadir Celik - "Robot"

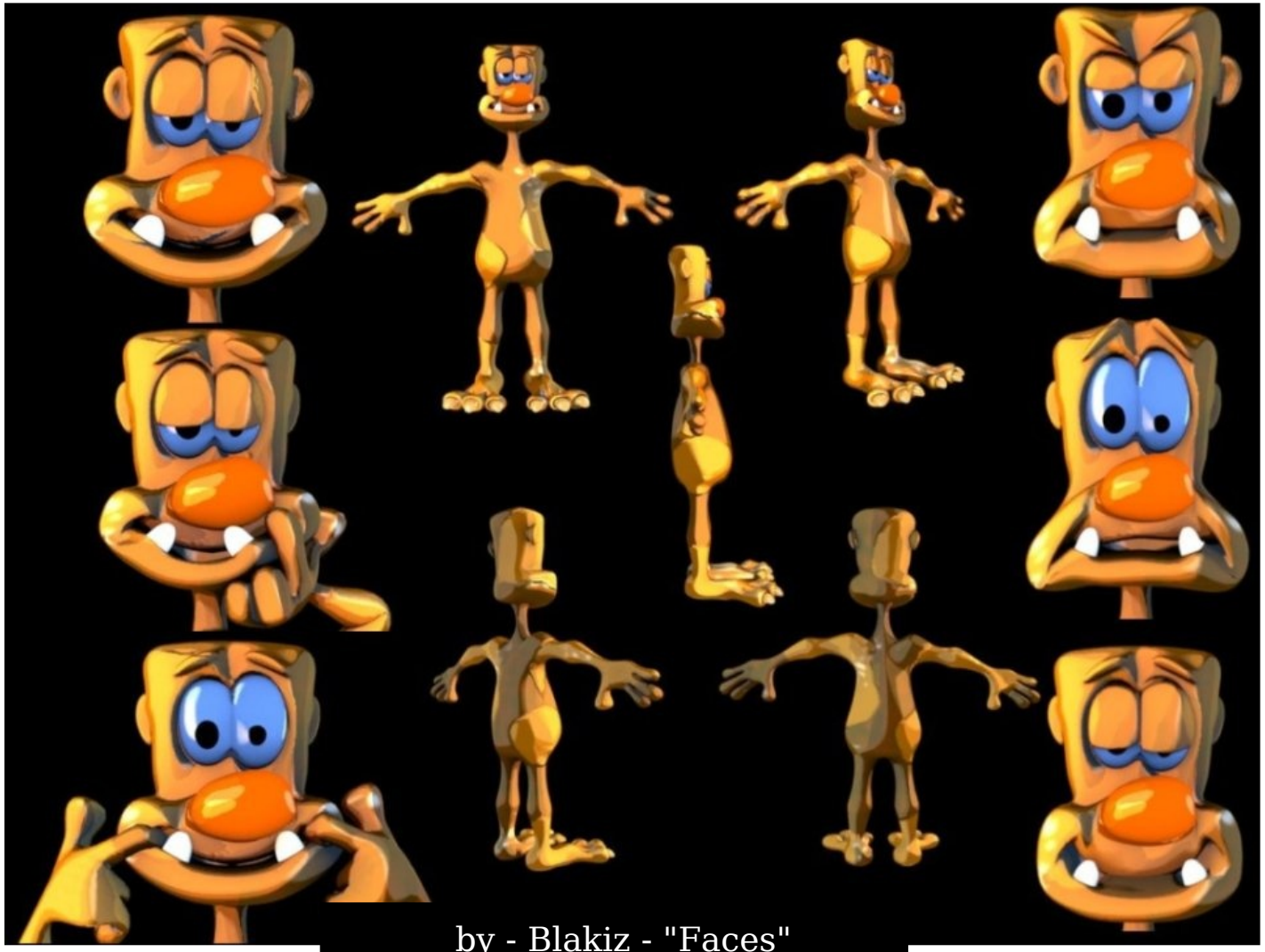
by - Timo - "Africa Mask"

by - Zooly - "The Princess 2"

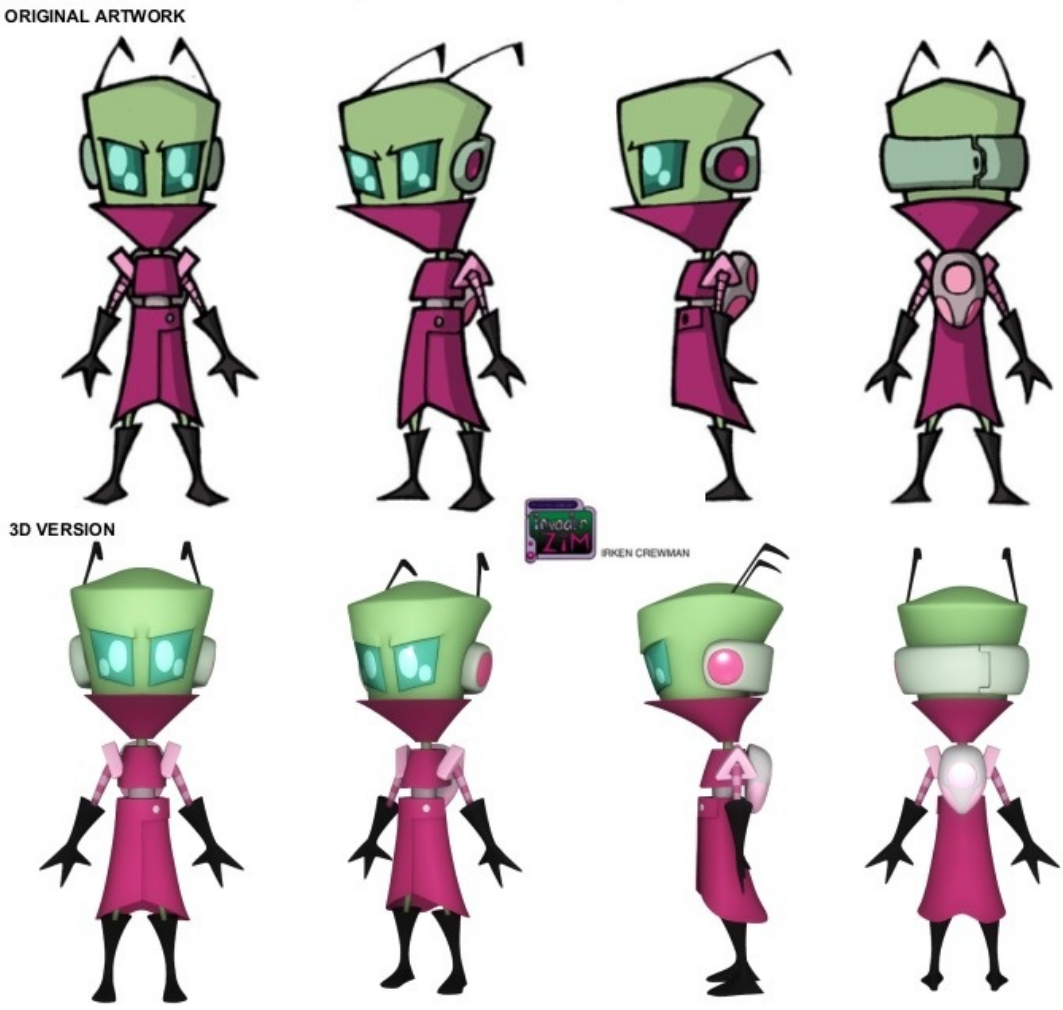
by - Zooly - "The Princess"

by - Zooly - "There is no limit"

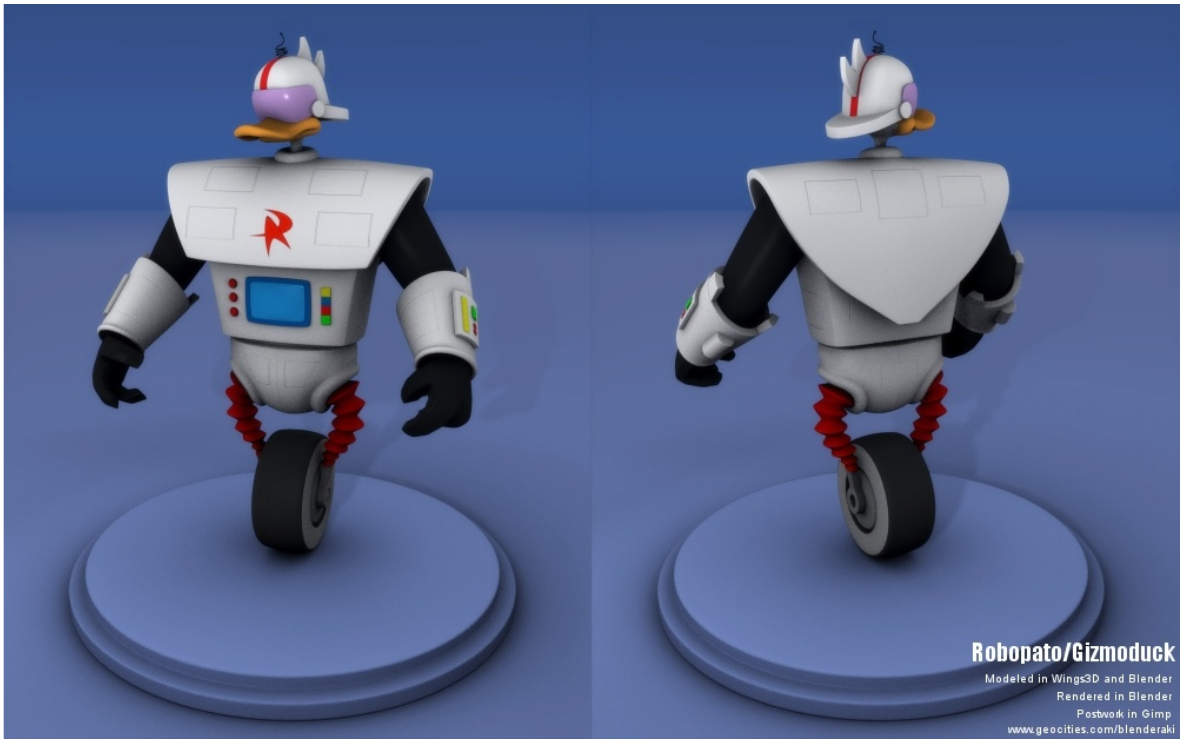
by - Adrea Giro - "Dino"

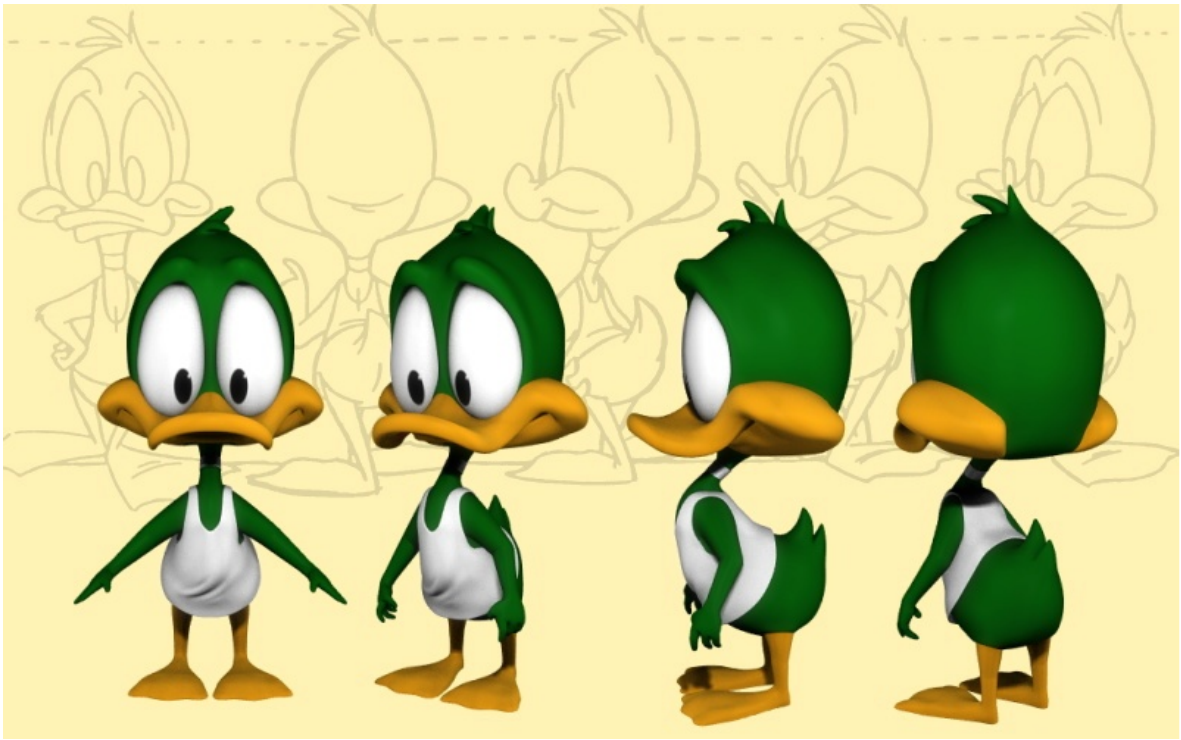


by - Blakiz - "Faces"



by - Blender aki - "BelaKiss" & "Irken Crewman"

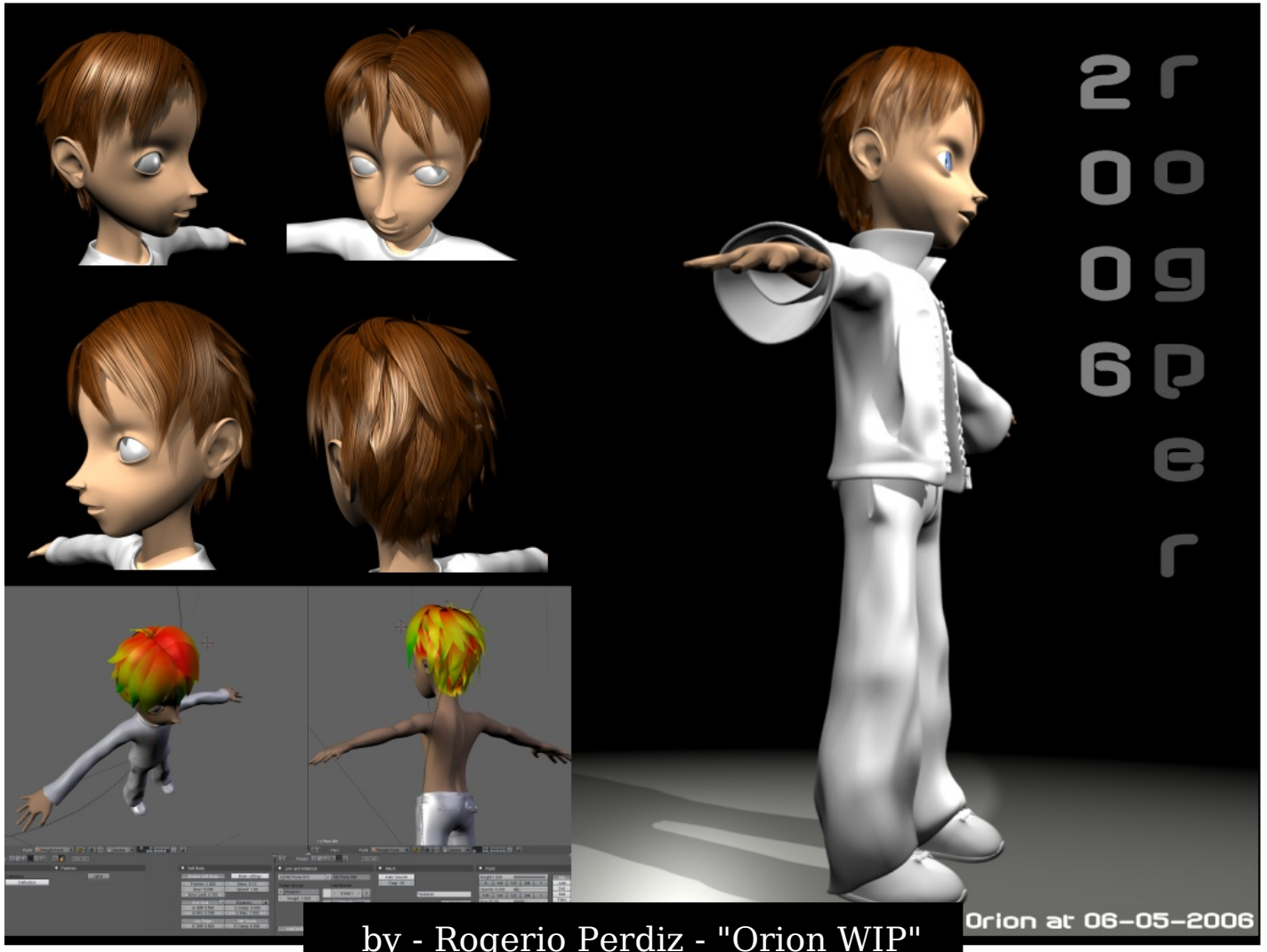




by - Blender aki - "Robopato" & "Plucky"

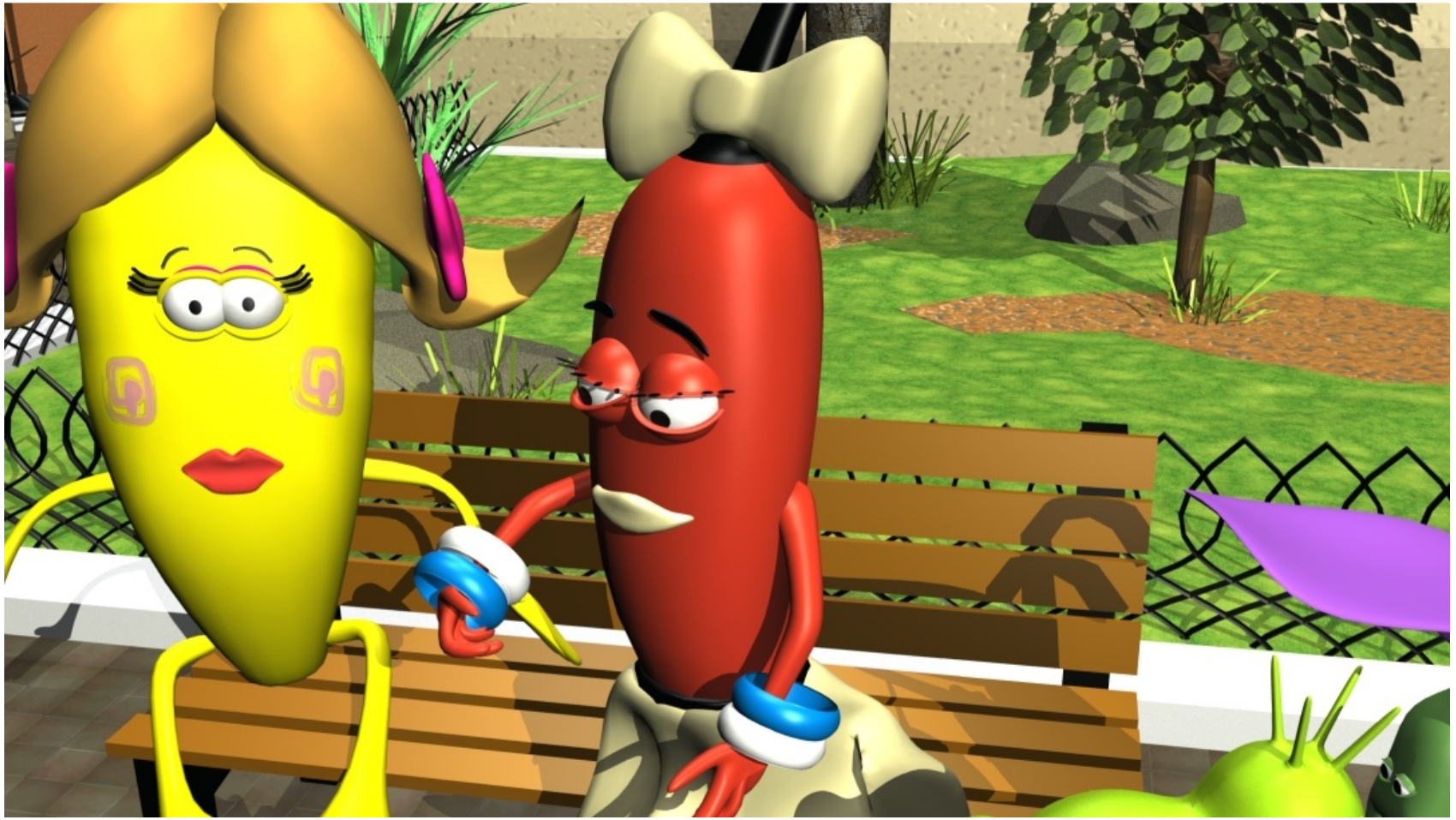
by - Luis Del- "Aguila head"

by - Caleb Ephratah - "The Knight"



by - Rogerio Perdiz - "Orion WIP"

by - Rogerio Perdiz - "Orion WIP"



by - Salvador gb Perdiz - "Mientos Chilmenton"

by - Simon Baudry - "Cimetierre"

by - Simon Baudry - "Meduse"

by - Rich - "Masked"

by - Zach Goldstein - "Lictor"

by - Tony Mullen - "The old man & the bean"

by - Roja - "Goblin male & Goblin female"

by - Roja - "Orc male & Orc female"

by - Roja - "Human female & "Texture block"

by - Roja - "Skeleton armed"

Farmer

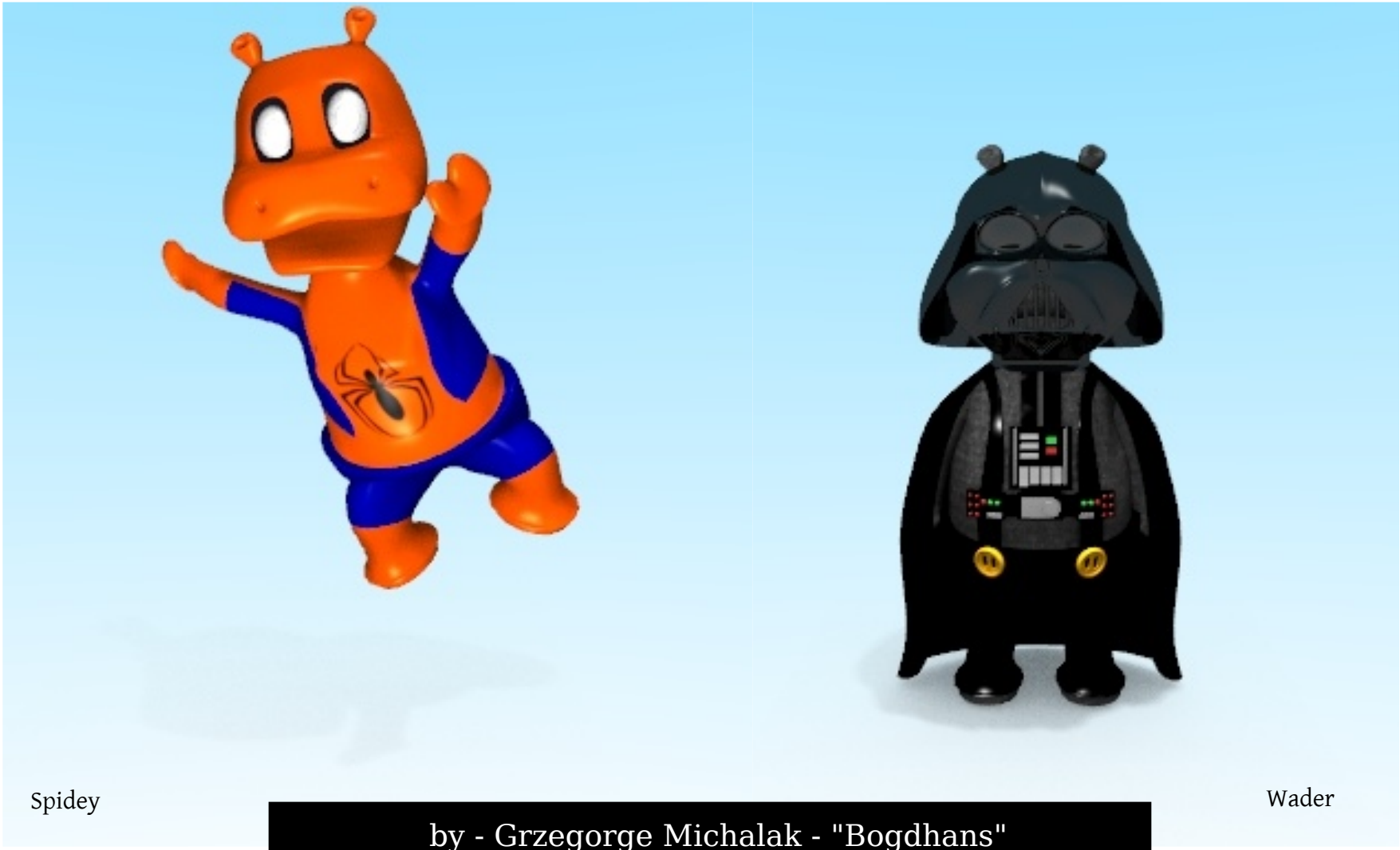
Matrix

by - Grzegorge Michalak - "Bogdhans"

Matrix2

Raper

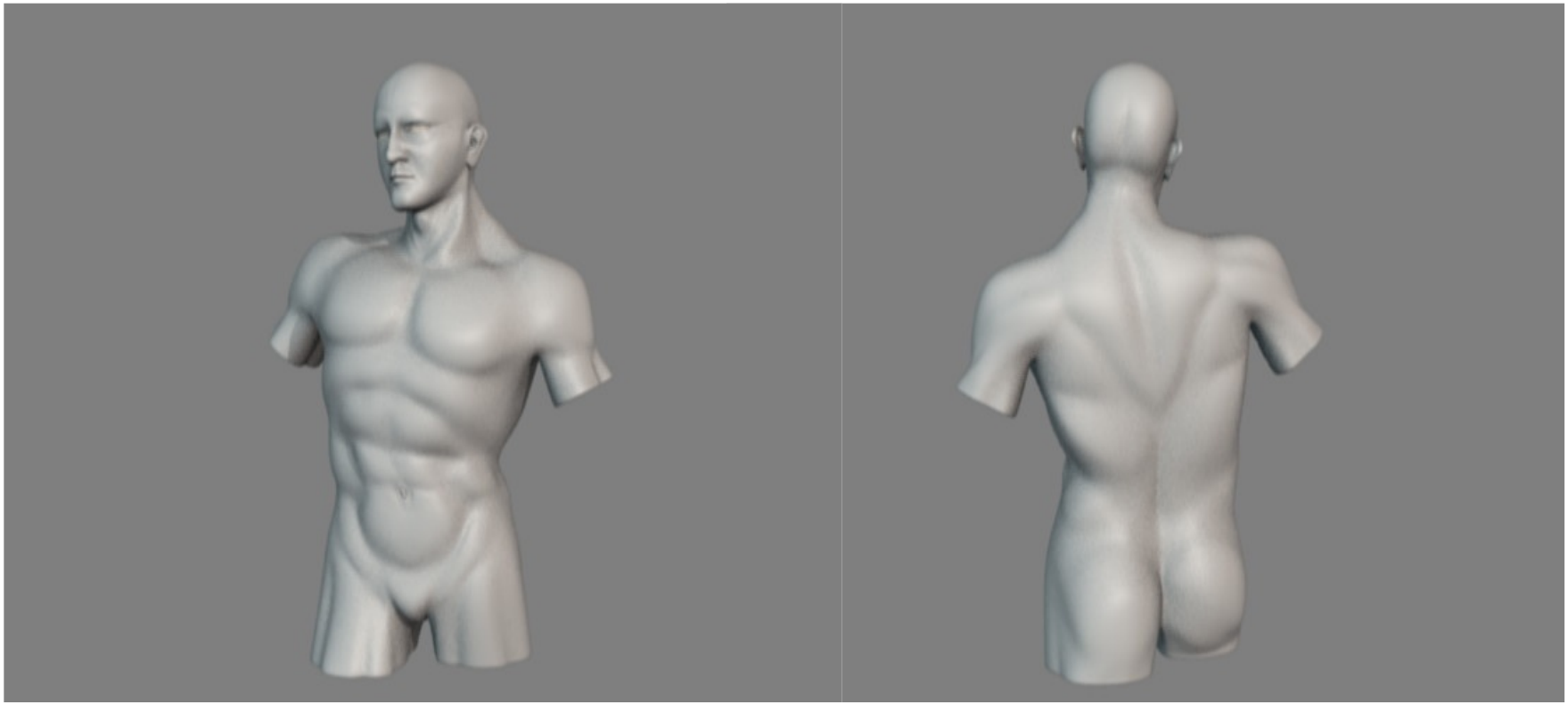
by - Grzegorge Michalak - "Bogdhans"



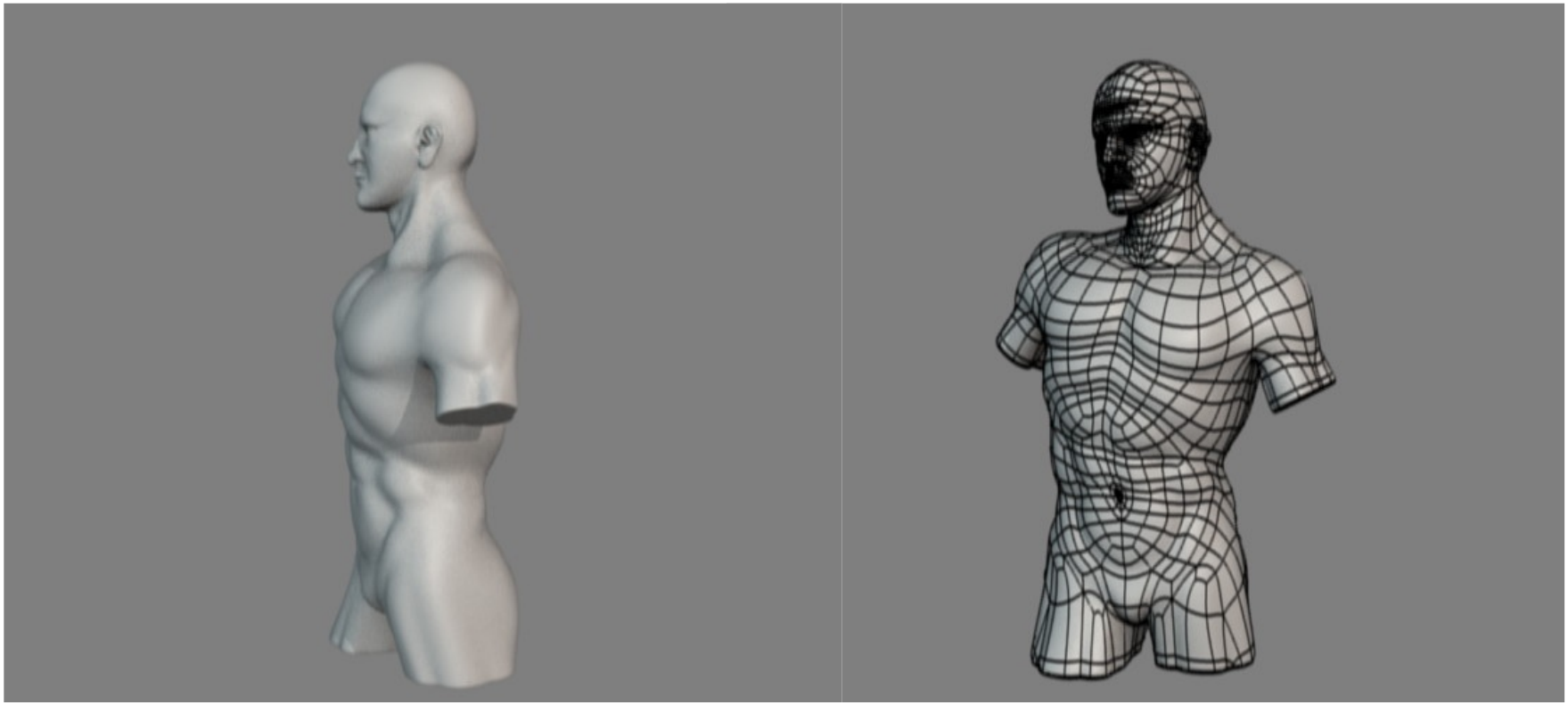
Spidey

Wader

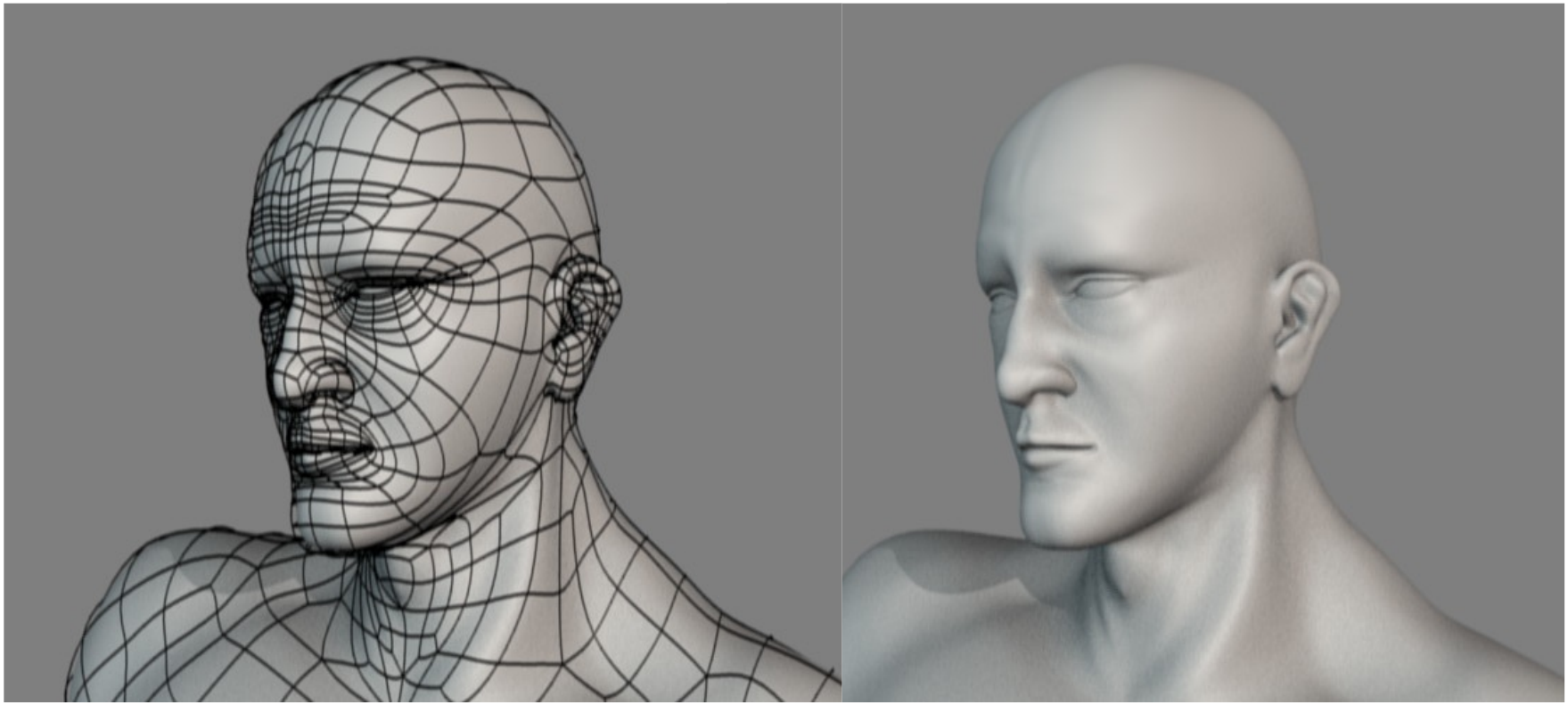
by - Grzegorge Michalak - "Bogdhans"



by - Jonatham Willimson - "Bust"



by - Jonatham Willimson - "Bust"



by - Jonatham Willimson - "Bust"

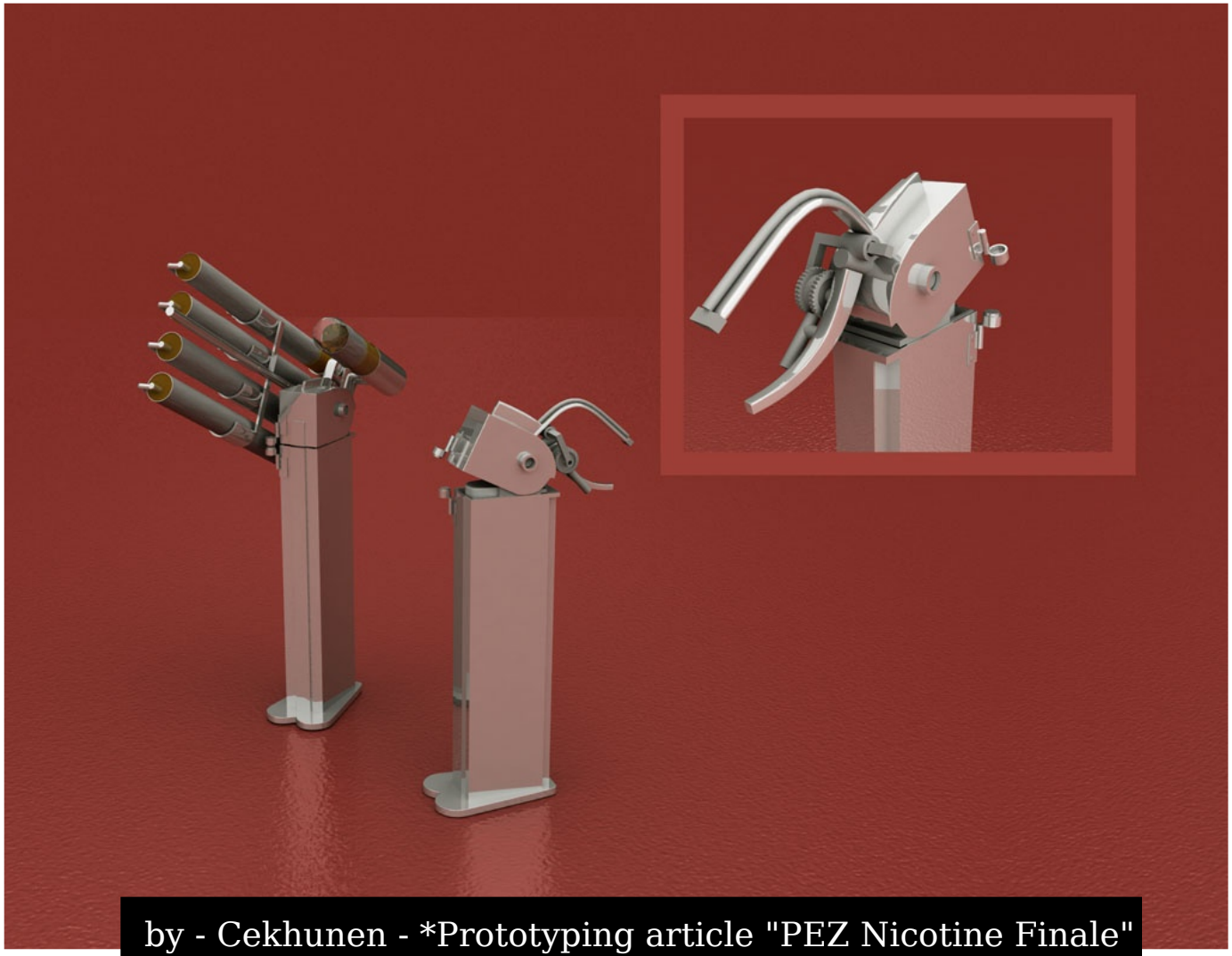
by - Cekhunen - *Prototyping article "Ashtray"

by - Cekhunen - *Prototyping article "Creamer Set"

by - Cekhunen - *Prototyping article "Sugar Full"

by - Cekhunen - *Prototyping article "PEZ Advile"

by - Cekhunen - *Prototyping article "PEZ Exactly Finale"



by - Cekhunen - *Prototyping article "PEZ Nicotine Finale"

Blenderart Staff

Gaurav Nawani	- Editor/Designer
Sandra Gilbert	- Managing editor/Proofer
Nam Pham	- Web Administrator/Proofer
Kenron Dillon	- Proof reader

Blenderart Contributors

Butterfly tutorial
Christian Guckelsberger
http://www.abyi.com/

Makehuman
Manuel Bastioni (English Version)
Alessandro Proglio
Antonio Di Cecca
Giovanni Lanza
Martin Mackinlay
http://sourceforge.net/projects/makehuman/

Character modeling at Plumiferos
Manuel Perez (Picasus)
Claudio Andaur (malefico)
http://www.plumiferos.com/

Blender2pov
José Mauricio
Rodas R. (Morfeus)

Rapid Prototyping with Blender
Claas Eicke Kuhnen (F.ip2)
www.ckbrd.de
www.concolori.de

Character Design 2d sketch to 3d
Rogério Perdiz (rogper)

Book Review - 3D Photorealism Toolkit
Ed Cicka

Respower
Cory King
Early Ehlinger
http://www.respower.com/

Issue5 available in July 2006

Theme: Modeling tricks and Blender Scripts

Articles : Using Blender for 3d game assests
Various modeling approaches
Extending your Blender via Scripts

Disclaimer